HK32F103x8T6A/HK32F103xBT6A

用户手册

Rev1.1.3

History

Version	Date	Description
1.0.0	2018/06/08	初始版本
1.0.1	2018/08/14	章节重新排序
1.0.2	2019/06/10	更正 FLASH 擦写次数为 1K
1.0.3	2019/06/10	更改型号 HK32F103XXT7-> HK32F103XXT6
1.0.4	2019/08/14	公司 logo 更新 页码更新
1.0.5	2019/08/14	更新 章节链接
1.0.6	2019/08/15	更新第 3 章、第 4 章、第 8 章
1.0.7	2019/08/16	10、11、12、20 章更新
1.0.8	2019/08/20	更新 USART 章节
1.0.9	2019/08/22	更新了 6，7，15，21 章
1.1.0	2019/08/23	更新了 5,9,13,14,16,17,18,19 章的寄存器和文档的格式
1.1.1	2020/04/07	去掉 UART4/5，增加 21 章节
1.1.2	2020/5/6	更新 7 章节时钟树
1.1.3	2020/8/21	更新公司 LOGO

说明

本文档为 HK32F103 C8T6A/CBT6A/R8T6A/RBT6A 芯片用户手册。

HK32F103 C8T6A/CBT6A/R8T6A/RBT6A 芯片是深圳市航顺芯片技术研发有限公司开发的低功耗 MCU 芯片，请联系深圳市航顺芯片技术研发有限公司提供更多相关文档。

目录

表清单

图清单

HK32 系列产品命名规则

示例： HK32 F 103 C 8 T 6 A xxx

产品系列

HK32 = 基于ARM®核心的32位微控制器

产品类型

F = 通用类型

产品子系列

101 = 基本型
102 = USB基本型，USB 2.0全速设备
103 = 增强型
105或107 = 互联型

引脚数目

T = 36脚
C = 48脚
R = 64脚
V = 100脚
Z = 144脚

闪存存储器容量

4 = 16K字节的闪存存储器
6 = 32K字节的闪存存储器
8 = 64K字节的闪存存储器
B = 128K字节的闪存存储器
C = 256K字节的闪存存储器
D = 384K字节的闪存存储器
E = 512K字节的闪存存储器

封装

H = BGA
T = LQFP
U = VFQFPN
Y = WLCSP64

温度范围

6 = 工业级温度范围，-40° C~85° C
7 = 工业级温度范围，-40° C~105° C

内部代码

A 或者空 (详见产品数据手册)

选项

xxx = 已编程的器件代号(3 个数字)
TR = 卷带式包装

1 文中的缩写

本文档为 HK32F103 芯片参考手册。
HK32F103 系列芯片是深圳市航顺芯片技术研有限公司旗下公司深圳市浩瀚天际处理器有限公司开发的低功耗 MCU 芯片，请联系深圳市浩瀚天际处理器有限公司提供更多相关文档支持设计开发。

1.1 寄存器描述表中使用的缩写列表

在对寄存器的描述中使用了下列缩写：

read / write (rw)	软件能读写此位。
read-only (r)	软件只能读此位。
write-only (w)	软件只能写此位，读此位将返回复位值。
read/clear (rc_w1)	软件可以读此位，也可以通过写'1'清除此位，写'0'对此位无影响。
read / clear (rc_w0)	软件可以读此位，也可以通过写'0'清除此位，写'1'对此位无影响。
read / clear by read (rc_r)	软件可以读此位；读此位将自动地清除它为'0'，写'0'对此位无影响。
read / set (rs)	软件可以读也可以设置此位，写'0'对此位无影响。
read-only write trigger (rt_w)	软件可以读此位；写'0'或'1'触发一个事件但对此位数值没有影响。
toggle (t)	软件只能通过写'1'来翻转此位，写'0'对此位无影响。
Reserved(Res.)	保留位，必须保持默认值不变

1.2 术语表

- 闪存存储器容量在 64K 至 128K 字节之间的 HK32F103xx 微控制器属于**中容量产品**。

1.3 可用的外设

有关 HK32 微控制器系列全部型号中，某外设存在与否及其数目，请查阅相应的 HK32F103xx 数据手册。

2 存储器和总线构架

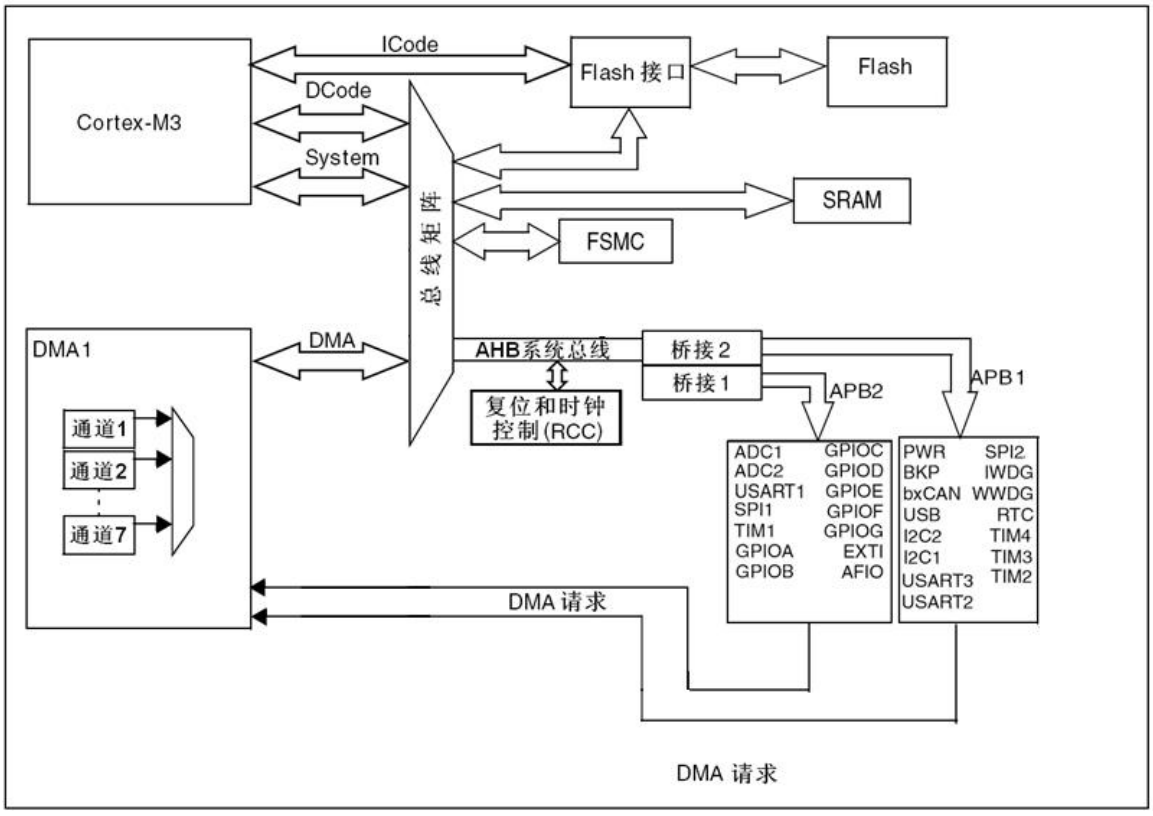
2.1 系统构架

主系统由以下部分构成：

- 四个驱动单元：
  - Cortex™-M3 内核 DCode 总线(D-bus)，和系统总线(S-bus)
  - 通用 DMA1
- 四个被动单元
  - 内部 SRAM
  - 内部闪存存储器
  - AHB 到 APB 的桥(AHB2APBx)，它连接所有的 APB 设备

这些都是通过一个多级的 AHB 总线构架相互连接的，如下图图 1 所示：

图 1 系统结构



ICode 总线

该总线将 Cortex™-M3 内核的指令总线与闪存指令接口相连接。指令预取在此总线上完成。

DCode 总线

该总线将 Cortex™-M3 内核的 DCode 总线与闪存存储器的数据接口相连接(常量加载和调试访问)。

系统总线

此总线连接 Cortex™-M3 内核的系统总线(外设总线)到总线矩阵，总线矩阵协调着内核和 DMA 间的访问。

DMA 总线

此总线将DMA的AHB主控接口与总线矩阵相联，总线矩阵协调着CPU的DCode和DMA到 SRAM、闪存和外设的访问。

总线矩阵

总线矩阵协调内核系统总线和 DMA 主控总线之间的访问仲裁，仲裁利用轮换算法。总线矩阵包含 4 个驱动部件(CPU 的 DCode、系统总线和 DMA1 总线)和 4 个被动部件(闪存存储器接口(FLITF)、SRAM 和 AHB2APB 桥)。

AHB 外设通过总线矩阵与系统总线相连，允许 DMA 访问。

AHB/APB 桥(APB)

两个 AHB/APB 桥在 AHB 和 2 个 APB 总线间提供同步连接。APB1 操作速度限于 48MHz，APB2 操 作于全速(最高 96MHz)。

有关连接到每个桥的不同外设的地址映射请参考表 1。在每一次复位以后，所有除 SRAM 和 FLITF 以外的外设都被关闭，在使用一个外设之前，必须设置寄存器 RCC_AHBENR、RCC_APB1ENR 和 RCC_APB2ENR 来打开该外设的时钟。

注意：当对 APB 寄存器进行 8 位或者 16 位访问时，该访问会被自动转换成 32 位的访问：桥会自动将 8 位或者 32 位的数据扩展以配合 32 位的向量。

2.2 存储器组织

程序存储器、数据存储器、寄存器和输入输出端口被组织在同一个 4GB 的线性地址空间内。数据字节以小端格式存放在存储器中。一个字里的最低地址字节被认为是该字的最低有效字节，而最高地址字节是最高有效字节。

外设寄存器的映像请参考相关章节。可访问的存储器空间被分成 8 个主要块，每个块为 512MB。

其他所有没有分配给片上存储器和外设的存储器空间都是保留的地址空间，请参考相应器件的数据手册中的存储器映像图。

2.3 存储器映像

请参考相应器件的数据手册中的存储器映像图。表 1 列出了所用 HK32F10xxx 中内置外设的起始地址。表 1 寄存器组起始地址

起始地址	外设	总线	寄存器映像
0x5000 0000 – 0x5003 FFFF	保留	AHB	
0x4003 0400 – 0x4FFF FFFF	保留		
0x4003 0000 – 0x4003 03FF	ALU 协处理器		参见 21 节
0x4002 8000 – 0x4002 FFFF	保留		
0x4002 3400 - 0x4002 3FFF	保留		
0x4002 3000 - 0x4002 33FF	CRC		参见 5.4.4 节
0x4002 2000 - 0x4002 23FF	闪存存储器接口		
0x4002 1400 - 0x4002 1FFF	保留		
0x4002 1000 - 0x4002 13FF	复位和时钟控制(RCC)		参见 8.3.11 节
0x4002 0800 - 0x4002 0FFF	保留		
0x4002 0400 - 0x4002 07FF	保留		
0x4002 0000 - 0x4002 03FF	DMA1		参见 12.4.7 节
0x4001 8400 - 0x4001 7FFF	保留		
0x4001 8000 - 0x4001 83FF	保留		
0x4001 4000 - 0x4001 7FFF	保留	APB2	
0x4001 3C00 - 0x4001 3FFF	保留		
0x4001 3800 - 0x4001 3BFF	USART1		参见 22.6.8 节
0x4001 3400 - 0x4001 37FF	保留		
0x4001 3000 - 0x4001 33FF	SPI1		参见 20.4 节
0x4001 2C00 - 0x4001 2FFF	TIM1 定时器		参见 13.4.21 节
0x4001 2800 - 0x4001 2BFF	ADC2		参见 11.12.15 节
0x4001 2400 - 0x4001 27FF	ADC1		参见 11.12.15 节
0x4001 2000 - 0x4001 23FF	保留		参见 9.5 节
0x4001 1C00 - 0x4001 1FFF	保留		参见 9.5 节
0x4001 1800 - 0x4001 1BFF	保留		参见 9.5 节
0x4001 1400 - 0x4001 17FF	GPIO 端口 D		参见 9.5 节
0x4001 1000 - 0x4001 13FF	GPIO 端口 C		参见 9.5 节
0X4001 0C00 - 0x4001 0FFF	GPIO 端口 B		参见 9.5 节
0x4001 0800 - 0x4001 0BFF	GPIO 端口 A		参见 9.5 节
0x4001 0400 - 0x4001 07FF	EXTI		参见 10.3.7 节
0x4001 0000 - 0x4001 03FF	AFIO		参见 9.5 节
0x4000 7800 - 0x4000FFFF	保留	APB1	
0x4000 7400 - 0x4000 77FF	保留		
0x4000 7000 - 0x4000 73FF	电源控制(PWR)		参见 6.4.3 节
0x4000 6C00 - 0x4000 6FFF	后备寄存器(BKP)		参见 7.4.5 节
0x4000 6800 - 0x4000 6BFF	保留		
0x4000 6400 - 0x4000 67FF	bxCAN		参见 19.9.5 节
0x4000 6000[(1)] - 0x4000 63FF	USB/CAN 共享的 512 字节 SRAM		

0x4000 5C00 - 0x4000 5FFF	USB 全速设备寄存器		参见 18.5.4 节
0x4000 5800 - 0x4000 5BFF	I2C2		参见 21.6.10 节
0x4000 5400 - 0x4000 57FF	I2C1		参见 21.6.10 节
0x4000 5000 - 0x4000 53FF	保留		
0x4000 4C00 - 0x4000 4FFF	保留		
0x4000 4800 - 0x4000 4BFF	USART3		参见 22.6.8 节
0x4000 4400 - 0x4000 47FF	USART2		参见 22.6.8 节
0x4000 4000 - 0x4000 3FFF	保留		
0x4000 3C00 - 0x4000 3FFF	保留		
0x4000 3800 - 0x4000 3BFF	SPI2		参见 20.4 节
0x4000 3400 - 0x4000 37FF	保留		
0x4000 3000 - 0x4000 33FF	独立看门狗(IWDG)		参见 16.4.4 节
0x4000 2C00 - 0x4000 2FFF	窗口看门狗(WWDG)		参见 17.6.4 节
0x4000 2800 - 0x4000 2BFF	RTC		参见 15.4.7 节
0x4000 1800 - 0x4000 27FF	保留		
0x4000 1400 - 0x4000 17FF	保留		
0x4000 1000 - 0x4000 13FF	保留		
0x4000 0C00 - 0x4000 0FFF	保留		
0x4000 0800 - 0x4000 0BFF	TIM4 定时器		参见 14.4.19 节
0x4000 0400 - 0x4000 07FF	TIM3 定时器		参见 15.4.19 节
0x4000 0000 - 0x4000 03FF	TIM2 定时器		参见 14.4.19 节

2.3.1 嵌入式 SRAM

HK32F10xxx 内置 64K 字节的静态 SRAM。它可以以字节、半字(16 位)或全字(32 位)访问。SRAM 的起始地址是 0x2000 0000。

2.3.2 位段

Cortex™-M3 存储器映像包括两个位段(bit-band)区。这两个位段区将别名存储器区中的每个字映射到位段存储器区的一个位，在别名存储区写入一个字具有对位段区的目标位执行读-改-写操作的相同效果。

在 HK32F10xxx 里，外设寄存器和 SRAM 都被映射到一个位段区里，这允许执行单一的位段的写和读操作。

下面的映射公式给出了别名区中的每个字是如何对应位带区的相应位的：

bit_word_addr = bit_band_base + (byte_offset×32) + (bit_number×4)

其中：

bit_word_addr 是别名存储器区中字的地址，它映射到某个目标位。

bit_band_base 是别名区的起始地址。

byte_offset 是包含目标位的字节在位段里的序号

bit_number 是目标位所在位置(0-31)

例子：

下面的例子说明如何映射别名区中 SRAM 地址为 0x20000300 的字节中的位 2：

0x22006008 = 0x22000000 + (0x300×32) + (2×4).

对 0x22006008 地址的写操作与对 SRAM 中地址 0x20000300 字节的位 2 执行读-改-写操作有着相同的效果。

读 0x22006008 地址返回 SRAM 中地址 0x20000300 字节的位 2 的值(0x01 或0x00)。

请参考《Cortex™-M3 技术参考手册》以了解更多有关位段的信息。

2.3.3 嵌入式闪存

高性能的闪存模块有以下的主要特性：

- 高达 128K 字节闪存存储器结构：闪存存储器有主存储块和信息块组成：
  - 主存储块容量：

    主存储块最大为 32K×32 位，每个存储块划分为 128 个 1K 字节的页(见表 2)。
  - 信息块容量：

    2560 字节(见表 2)。

闪存存储器接口的特性为：

- 带预取缓冲器的读接口
- 选择字节加载器
- 闪存编程/擦除操作
- 访问/写保护

表 2 闪存模块的组织

模块	名称	地址	大小(字节)
主存储块	页 0	0x0800 0000 - 0x0800 03FF	1K
	页 1	0x0800 0400 - 0x0800 07FF	1K
	页 2	0x0800 0800 - 0x0800 0BFF	1K
	页 3	0x0800 0C00 - 0x0800 0FFF	1K
	页 4	0x0800 1000 - 0x0800 13FF	1K
	… …	… …	… …
	页 127	0x0801 FC00 - 0x0801 FFFF	1K
信息块	系统存储器	0x1FFF F000 - 0x1FFF F7FF	2K
	选择字节	0x1FFF F800 - 0x1FFF F9FF	512
闪存存储器接口寄存器	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FALSH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	保留	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4
	FLASH_ECR	0x4002 2024 - 0x4002 2027	4

闪存读取

闪存的指令和数据访问是通过 AHB 总线完成的。预取模块是用于通过 ICode 总线读取指令的。仲裁是作用在闪存接口，并且 DCode 总线上的数据访问优先。

读访问可以有以下配置选项：

- 等待时间：可以随时更改的用于读取操作的等待状态的数量。
- 预取缓冲区：在每一次复位以后被自动打开，由于预取缓冲区的存在，CPU 可以工作在更高的主频。CPU 每次取指最多为 32 位的字，取一条指令时，下一条指令已经在缓冲区中等待。
- 半周期：用于功耗优化。

注： *1.这些选项应与闪存存储器的访问时间一起使用。等待周期体现了系统时钟(SYSCLK)频率与闪存访问时间的关系：*

0 等待周期，当 0 < HCLK < 24MHz

1 等待周期，当 24MHz < HCLK ≤ 48MHz

2 等待周期，当 48MHz < HCLK ≤ 72MHz

3 等待周期，当 72MHz < HCLK ≤ 96MHz

2.半周期配置该特性只能用在 HCLK 时钟频率为 8MHz 或低于 8MHz 时。

3.使用 DMA：DMA 在 DCode 总线上访问闪存存储器，它的优先级比 ICode 上的取指高。DMA 在每次传送完成后具有一个空余的周期。有些指令可以和 DMA 传输一起执行。

编程和擦除闪存

闪存编程一次可以写入 16 位(半字)或者 32 位(字)。 闪存擦除操作可以按页面擦除、半页擦除或完全擦除(全擦除)。全擦除不影响信息块。 为了确保不发生过度编程，闪存编程和擦除控制器块是由一个固定的时钟控制的。 写操作(编程或擦除)结束时可以触发中断。仅当闪存控制器接口时钟开启时，此中断可以用来从 WFI 模式退出。

2.4 启动配置

在 HK32F10xxx 里，可以通过 BOOT[1:0]引脚选择三种不同启动模式。

表 3 启动模式

启动模式选择引脚		启动模式	说明
BOOT1	BOOT0		
X	0	主闪存存储器	主闪存存储器被选为启动区域
0	1	系统存储器	系统存储器被选为启动区域
1	1	内置 SRAM	内置 SRAM 被选为启动区域

在系统复位后，SYSCLK 的第 4 个上升沿，BOOT 引脚的值将被锁存。用户可以通过设置 BOOT1 和 BOOT0 引脚的状态，来选择在复位后的启动模式。

在从待机模式退出时，BOOT 引脚的值将被被重新锁存；因此，在待机模式下 BOOT 引脚应保持为需要的启动配置。在启动延迟之后，CPU 从地址 0x0000 0000 获取堆栈顶的地址，并从启动存储器的 0x0000 0004 指示的地址开始执行代码。

因为固定的存储器映像，代码区始终从地址 0x0000 0000 开始(通过 ICode 和 DCode 总线访问)，而数据区(SRAM)始终从地址 0x2000 0000 开始(通过系统总线访问)。Cortex-M3 的 CPU 始终从ICode 总线获取复位向量，即启动仅适合于从代码区开始(典型地从 Flash 启动)。HK32F10xxx 微控制器实现了一个特殊的机制，系统可以不仅仅从 Flash 存储器或系统存储器启动，还可以从内置 SRAM 启动。

根据选定的启动模式，主闪存存储器、系统存储器或 SRAM 可以按照以下方式访问：

- 从主闪存存储器启动：主闪存存储器被映射到启动空间(0x0000 0000)，但仍然能够在它原有的地址(0x0800 0000)访问它，即闪存存储器的内容可以在两个地址区域访问，0x0000 0000 或 0x0800 0000。
- 从系统存储器启动：系统存储器被映射到启动空间(0x0000 0000)，但仍然能够在它原有的地址(地址为 0x1FFF F000)访问它。
- 从内置 SRAM 启动：RAM 被映射到启动空间(0x0000 0000)，也能在 0x2000 0000 开始的地址区访问 SRAM。

内嵌的自举程序

内嵌的自举程序存放在系统存储区，在生产线上写入，用于通过可用的串行接口对闪存存储器进行重新编程：

- 可以通过 USART1 接口启用自举程序。进一步的 细节请参考 system memory boot mode 的手册。

3 嵌入式闪存

3.1 闪存主要特性

- 高达 128K 字节闪存存储器
- 存储器结构
  - 主闪存模块：32K 字（32K×32 位）
  - 信息模块：512 字（512×32 位）
  - 选项字节有 128 个字

闪存的接口特征

- 带预取缓冲器的读接口
- 选择字节加载器
- 闪存编程/擦除操作
- 访问/写保护
- 低功耗模式

3.2 闪存功能描述

3.2.1 闪存结构

闪存空间由 32 位宽的存储单元组成，既可以存代码又可以存数据。主闪存块有 128 页（每页 1K 字节）。模块如下图所示：

模块	名字	基地址	大小
主闪存储块	Page0	0x0800 0000-0x0800 03FF	1KB
	Page1	0x0800 0400-0x0800 07FF	1KB
	Page2	0x0800 0800-0x0800 0BFF	1KB
	Page3	0x0800 0C00-0x0800 0FFF	1KB
	Page4	0x0800 1000-0x0800 13FF	1KB
	……	……	……
	Page127	0x0801 FC00-0x0801 FFFF	1KB
信息模块	系统存储	0x1FFF F000-0x1FFF F7FF	2KB
	选择字节	0x1FFF F800-0x1FFF F9FF	16

闪存接口寄存器	FLASH_ACR	0x4002 2000-0x4002 2003	4
	FLASH_KEYR	0x4002 2004-0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008-0x4002 2003B	4
	FLASH_SR	0x4002 200C-0x4002 200F	4
	FLASH_CR	0x4002 2010-0x4002 2013	4
	FLASH_AR	0x4002 2014-0x4002 2017	4
	保留	0x4002 2018-0x4002 201B	4
	FLASH_OBR	0x4002 201C-0x4002 201F	4
	FLASH_WRPR	0x4002 2020-0x4002 2023	4
	FLASH_ECR	0x4002 2024 - 0x4002 2027	4

3.2.2 读操作

嵌入式 Flash 模块可以像普遍存储空间一样直接寻址访问。任何对 Flash 模块内容的读操作都须经过专门的判断过程。
取指令和取数据都是通过 AHB 总线读取访问。它主要的工作就是产生控制信号，然后读取闪存里面的信息和预取。预取模块仅运用在通过 I-Code 总线实现指令预取。D-Code 总线具有更高的预取优先级。

取指

Cortex-M3 通过 I-Code 和 D-Code 实现取指。预取旨在增加总线的效率。

预取缓冲区

预取指缓冲区的内容与 Flash 相同，能够完全替代一次同样大小的读取访问。预取缓冲区的工作使得更快速的 CPU 执行成为可能，因为 CPU 取一个字指令的同时下一个字的指令内容页已经在预取缓冲区中。

预取控制器

预取控制器会根据预取缓冲区的可用空间来把握访问 Flash 的时机。当预取缓冲区中存在至少一块可用空间时，预取控制器会发起一次读取请求。复位后，预取指缓冲区的默认状态是打开的。

访问等待周期

为了保护对 Flash 的正确读取，必须在 Flash 访问控制寄存器中的 LATENCY[2:0]中指定预取指控制器的速度比，这个数值等于每次访问 Flash 后到下次访问之间所需插入的等待周期的个数。复位后，这个值默认为零，也就是没有插入等待周期的状态。

D-Code 接口

D-Code 接口在 CPU 端包含一个简单的 AHB 接口和产生闪存访问控制的仲裁申请。D-Code 具有更高于预取的优先级。这个接口运用在预取缓存的访问时钟调谐器。

闪存访问控制

主要的，这个模块是一个简单的在预取(I-Code)读申请和 D-Code 间的仲裁。
D-Code 接口申请具有优于 I-Code 申请的优先级。

3.2.3 Flash 写和擦除操作

闪存擦写模块处理闪存的编程和擦除，它包含 8 个 32 位的寄存器。

烧写和擦除操作在整个产品工作电压范围内都可以完成。该操作由下列 8 个寄存器完成：

- 关键字寄存器（FLASH_KEYR）
- 选项字节关键字寄存器（FLASH_OPRKEYR）
- Flash 控制寄存器（FLASH_CR）
- Flash 状态寄存器（FLASH_SR）
- Flash 地址寄存器（FLASH_AR）
- 选项字节寄存器（FLASH_OBR）
- 写保护寄存器（FLASH_WRPR)
- Flash 控制寄存器 2 (FLASH_ECR)

只要 CPU 不去访问 Flash 空间，进行中的 Flash 写操作不会妨碍 CPU 的运行。也就是说，在对 Flash 进行写/擦除操作的同时，任何对 Flash 的访问都会令总线停顿，直到写/擦除操作完成后才会继续执行，这意味着在写/擦除 Flash 的同时不可以对它取指和访问数据。

关键值

这些关键值如下：

- RDPRT 值 = 0x00A5
- KEY1 = 0x45670123
- KEY2 = 0xCDEF 89AB

对 Flash 空间的解锁

复位后，Flash 存储器默认是受保护状态的，这样可以防范意外的擦除动作。FLASH_CR 寄存器不允许被改写，除非执行一串针对 FLASH_KEYR 寄存器的解锁操作才能开启对 FLASH_ CR 的访问权限。这串操作由下面 2 个写操作构成：

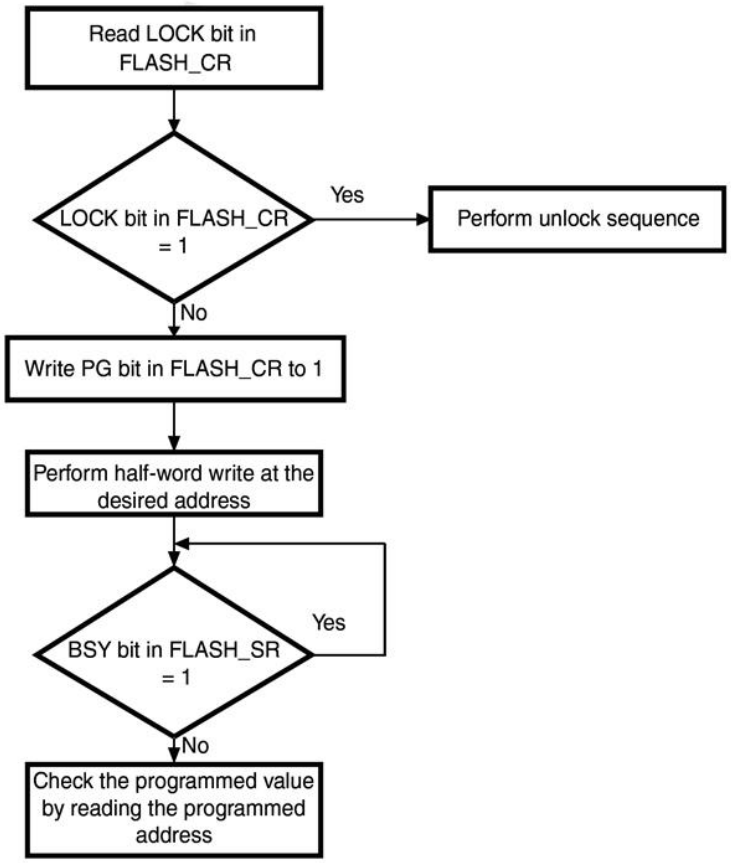
- 写关键字 KEY1=0x45670123
- 写关键字 KEY2=0xCDEF89AB

任何错误的顺序将会锁死 FLASH_CR 直至下次复位。当发生关键字错误时，会由总线错误引发一次硬件错误中断。如果 KEY1 出错就会立即中断，或如果 KEY1 正确但 KEY2 错误时就会在 KEY2 错的那个时候引发中断。

主闪存编程

主闪存一次可以编程 16 位或者 32 位，由 FLASH_CR 和 FLASH_ECR 决定。当 FLASH_CR 中的 PG 位为 1 时，直接对相应的地址写一个半字（16 位），就是一次编程操作。或者当 FLASH_ECR 中的 WPG 位为 1 时，直接对相应的地址写一个字（32 位），也是一次编程操作。如果 FLASH_CR 中的 PG 为 1 并试图写别的长度而不是半字，将引起总线错误中断。如果在编程的过程中读/写（BSY 位会被置 1），CPU 会被挂起，直到闪存编程完成。

半字编写流程如下图：



字编程的流程相似，区别在于是把 FLASH_ECR 中的 WPG 位置 1，而不是吧 FLASH_CR 中的 PG 位置 1。

标准编程

在这种模式 CPU 编程执行标准的半字写操作。FLASH_CR 中的 PG 位必须被置 1。

Flash 存储器接口会预读一下待编程地址的数据然后判断是否被擦除，如果不是，那么编程操作会自动取消，并且在 FLASH_SR 寄存器的 PGERR 位上提示编程错误告警。如果被编程的内容为全零，则会例外，这时会正确编程并且不告警。

如果待编程地址所对应的 FLASH_WRPR 中的写保护位有效，同样也不会有编程动作，同样也会产生编程错误告警。编程动作结束后，FLASH_SR 寄存器中得 EOP 位会给出提示。

主 Flash 存储器标准模式下的编程过程如下：

- 检查 FLASH_SR 中的 BSY 位，以确认上次操作已经结束
- 置位 FLASH_CR 寄存器中的 PG 位
- 根据配置，以半字为单位向目标地址写入数据
- 等待 FLASH_SR 寄存器中的 BSY 归零
- 读取编程的值然后验证

注意： 当 FLASH_SR 中的 BSY 被置 1 时，写模式下的寄存器不能被读。

Flash 存储器擦除

Flash 存储器可以按页或半页为单位擦除，也可以整片擦除。

页擦除

擦除页的步骤如下：

- 检查 FLASH_SR 中的 BSY 位，以确认上次操作已经结束
- 置位 FLASH_CR 寄存器中得 PER 位为 1
- 写 FLASH_AR 寄存器以选择待擦除的页
- 置位 FLASH_CR 寄存器中的 STRT 位为 1
- 等待 FLASH_SR 中的 BSY 归零

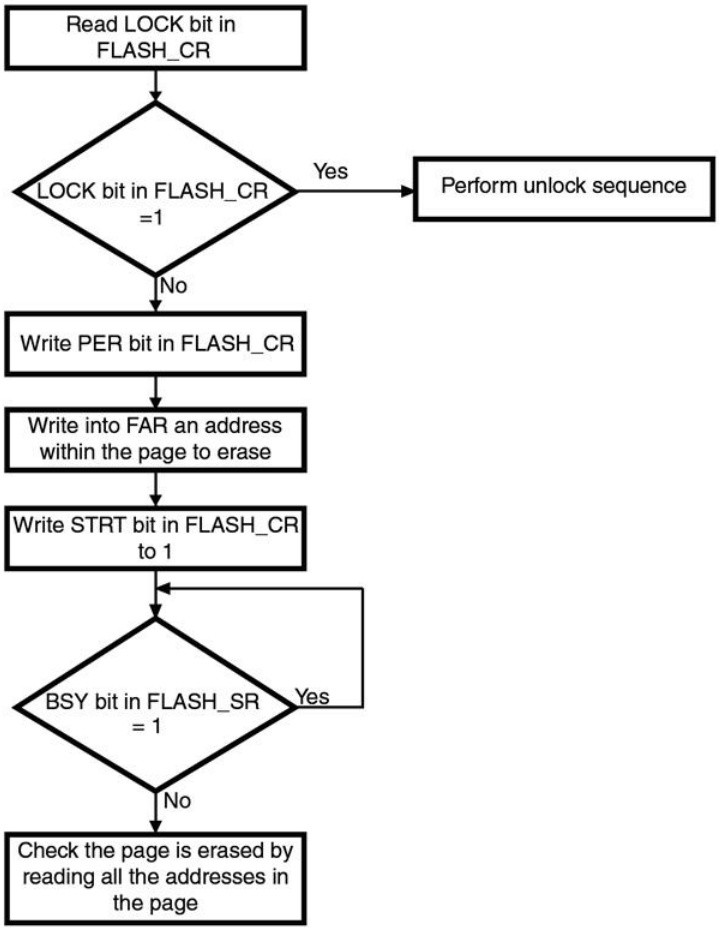
- 读擦除的页然后验证

半页擦除

擦除半页的步骤如下：

- 检查 FLASH_SR 中的 BSY 位，以确认上次操作已经结束
- 置位 FLASH_ECR 寄存器中得 HPER 位为 1
- 写 FLASH_AR 寄存器以选择待擦除的页
- 置位 FLASH_CR 寄存器中的 STRT 位为 1
- 等待 FLASH_SR 中的 BSY 归零
- 读擦除的页然后验证

擦除页流程如下图：



半页擦除

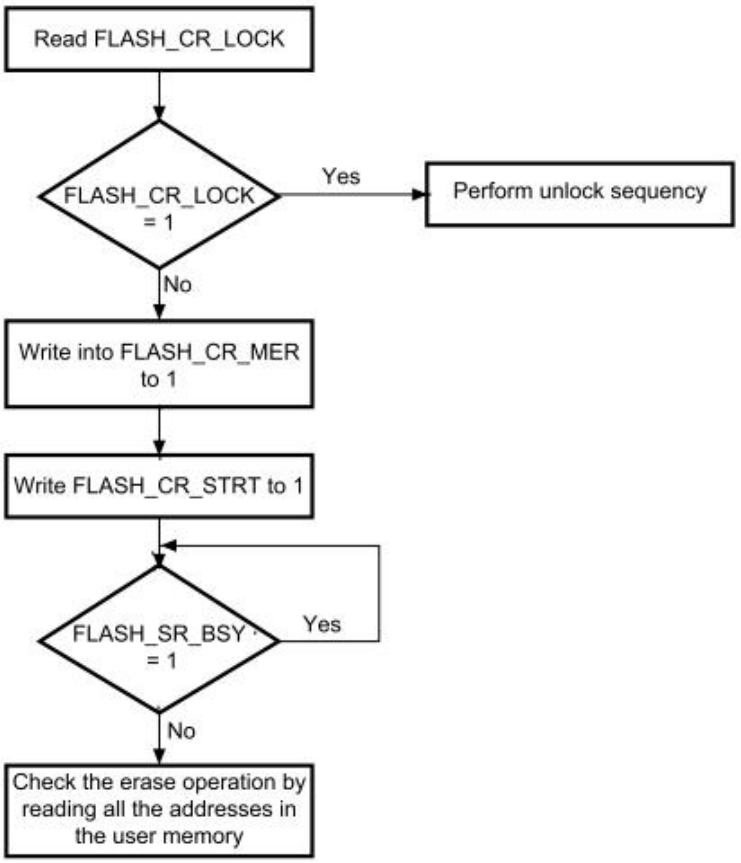
Flash 半页为 512 字节，半页擦除流程和页擦除类似，区别在于把 FLASH_ECR 中的 HPER 位置 1，而不是把 FLASH_CR 中的 PER 置 1。

整片擦除

可以用整片擦除命令一次擦除整个 Flash 户区，但信息块不会受这个命令影响，具体步骤如下：

- 检查 FLASH_SR 中的 BSY 位，以确认上次操作已经结束
- 置 FLASH_CR 寄存器中的 MER 位为 1
- 置 FLASH_CR 寄存器中的 STRT 位为 1
- 等待 BSY 位归零
- 读页里的所有值然后验证

擦除整页的流程如下图



选项字节编程

选项字节的编程与常规用户地址不同，总共就是 512 个字节（4 个写保护，1 个读保护，1 个硬件配置和 250 个用户数据储存）。解除 Flash 访问限制后，还需要针对 FLASH_OPTKEYR 寄存器完成关键字写入操作。完成该操作后，FLASH_CR 寄存器中的 OPTWRE 位会被置 1，然后就可以先置位 FLASH_ CR 中的 OPTPG 位，再按半字单位写目标地址。同样是会自动检查选项字节是否被擦除过，否则相关操作会被取消并且在 FLASH_SR 中的 PGERR 位提示错误。编程操作结束后，会由 FLASH_SR 寄存器的 EOP 位给出提示。

在编程操作开始前，把 LSB 值自动取反补到 MSB，这会保证选项字节的值总是对的。步骤如下：

- 检查 FLASH_SR 寄存器中的 BSY 位，以确保上次操作结束
- 解锁 FLASH_CR 寄存器中的 OPTWRE 位
- 置 FLASH_CR 寄存器中的 OPTPG 位为 1
- 写数据（半字）到目标地址
- 等待 BSY 位归零
- 读编程的值然后验证

当读保护选项字节由保护状态被改成非保护状态时，会自动引发一次整片擦除，然后才改写读保护位数据。如果用户只想改写其他的字节，则不会引发整片擦除，这个机制用于保护 Flash 的内容。

擦除过程

选项字节的擦除过程如下：

- 检查 FLASH_SR 寄存器中的 BSY 位，以确保上次操作结束
- 解锁 FLASH_CR 寄存器中的 OPTWRE 位
- 置 FLASH_CR 寄存器中的 OPTER 位为 1
- 置 FLASH_CR 寄存器中的 STRT 位为 1
- 等待 BSY 位归零
- 读取并校验

3.2.4 保护

可以防范用户区 Flash 区的代码被不可信的代码读出，也可以防范在程序跑飞的时候对 Flash 的意外擦除，写保护的最小单位是 4 页。

读保护

选项字节中的 RDP 字节置位，然后重新复位，读保护就被激活了。

注意： 如果读保护被设置在调试 JTAG/SWD 已连接的时候，POR 可以代替系统复位。

- 主片区 Flash 内容只能被用户程序读取（如果连接了 Debugger，用户程序也不能读取主片区 Flash 内容）
- 页 0~3 时自动写保护的，剩下的闪存空间可以被用户程序编程（写入 IAP, 常量储存等），这个保护只针对在调试模式和从 SRAM boot 时的擦除和编程，不保护整片擦除。
- 可以下载代码到 SRAM 和从 SRAM 中执行程序，它也可以被用来关闭读保护。当读保护选项字节被改变，会引发整片擦除。
- 当从 SRAM 启动的时候，使用 DMA 的闪存访问会被禁止
- 使用 JRAG,SWV(串行模式), SWD 和边界扫描时，闪存的访问被禁止

Flash 存储器的保护级别和RDP 选项字节及其补数内容的对应关系，如下表：

RDP 字节值	RDP 补码值	读保护级别
0xFF	0xFF	保护
RDPRT	RDP 反码	没有保护
其他值	非 RDP 反码值	保护

注意：擦除选项字节不会触发整页擦除，因为擦除值 0xFF 和保护值对应

读保护解除

从 SRAM 中解除读保护的方法：

- 擦除整个选项区间，结果时读保护代码 RDP 会被置为 0xFF, 此时读保护仍然可以被使能
- 写入正确的 RDP 值 0x00A5 去解除读保护，这个操作会引发一次主闪存的整片擦除
- POR 复位会重载选项字节 RDP 从而关闭读保护

注意：bootloader 可以关闭读保护（在这种情况下只有系统复位能重载选项字节）

3.2.5 写保护

写保护以 4 页为单位来控制，配置选项字节中的 WRP 位，然后复位是自动重新加载选项字节就可以使能这个保护。如果试图写入或擦除一个受保护的扇区，会引起 FLASH_SR 中的 WRPRTERR 标志位被置位。

写保护的解除

解除写保护有 2 个应用例子可以提供：

- 例 1：在解除写保护后禁止读保护
  - 使用 FLASH_CR 中的 OPTER 位擦除整个选项字节区域
  - 向 RDP 写入 0xA5 从而解除所有保护，这自然会引起整片擦除
  - 复位引起选项字节重新加载，写保护解除
- 例 2：在解除写保护后，读保护仍然有效，这种办法对使用用户 boot loader 进行在应用编程时很有用
  - 使用 FLASH_CR 中的 OPTER 位擦除整个选项字节区域
  - 复位引起选项字节重新加载，写保护解除

3.2.6 选项字节的写保护

选项字节默认是写保护的并且任何时候都可读。为了对选项字节进行写/擦除操作，必须对 POTKEYR 顺序写入关键字。正确的关键字会引起 FLASH_CR 中的 OPTWRE 置位，表明解锁成功。同样，通过对该位清零，能够再度禁止对选项字节的写操作。

3.3 选项字节说明

选项字节由用户根据应用的需要配置；例如：可以选择使用硬件模式的看门狗或软件的看门狗。 在选项字节中每个 32 位的字被划分为下述格式：

选项字节格式：

位 31：24	位 23：16	位 15：8	位 7：0
选项字节 1 的反码	选项字节 1	选项字节 0 的反码	选项字节 0

注意： 反码由硬件自动实现，软件写无效

选项字节块中选项字节的组织结构如下表所示。 选项字节可以从下表列出的存储器地址读出，或从选项字节寄存器（FLASH_OBR）读出。

注：新写入的选项字节（用户的或读/写保护的），在系统复位后才生效。

选项字节结构

地址	[31:24]	[23:16]	[15:8]	[7:0]
0x1FFF F800	nUSER	USER	nRDP	RDP
0x1FFF F804	nData1	Data1	nData0	Data0
0x1FFF F808	nWRP1	WRP1	nWRP0	WRP0
0x1FFF F80C	nWRP3	WRP3	nWRP2	WRP2
0x1FFF F810 ~ 0x1FFF F9FF	nDATA	DATA	nDATA	DATA

选项字节说明

存储器地址	选项字节
0x1FFF F800	位 [31：24] nUSER 位 [23：16]USER：用户选项字节（保存在 FLASH_OBR[9：2] 中）。这个字节用于配置下列 功能： 选择看门狗事件：硬件或软件 进入停机（STOP）模式时触发复位 进入待机模式时触发复位 注：只使用位 [16：18]，位[23：19]未使用。 位 18：nRST_STDBY 0：当进入待机模式时产生复位 1：进入待机模式时不产生复位 位 17：nRST_STOP 0：当进入停机（STOP）模式时产生复位 1：进入停机（STOP）模式时不产生复位 位 16：WDG_SW 0：硬件看门狗 1：软件看门狗 位 [15：8] nRDP 位 [7：0] nRDP：读保护选项字节 读保护帮助用户保护存储在 Flash 内部的程序，可以通过设置 RDP 选项字节来使能读保护。当 RDP 选择字配置值非 0xA5 时，使能读保护。读保护状态存储在 FLASH_OBR[1]。

0x1FFF F804	**Datax**：2 个字节的用户数据 这个地址可以使用选项字节的编程方式编程。 位 [31：24]：nData1 位 [23：16]：Data1(存储在 FLASH_OBR[25：18]) 位 [15：8]：nData0 位 [7：0]：Data0(存储在 FLASH_OBR[17：10])
0x1FFF F808	**WRPx**：闪存写保护选项字节 位 [31：24]：nWRP1 位 [23：16]：WRP1(存储在 FLASH_WRPR[15：8]) 位 [15：8]：nWRP0 位 [7：0]：WRP0(存储在 FLASH_WRPR[7：0])
0x1FFF F80C	**WRPx**：闪存写保护选项字节 位 [31：24]：nWRP3 位 [23：16]：WRP3(存储在 FLASH_WRPR[31：24]) 位 [15：8]：nWRP2 位 [7：0]：WRP2(存储在 FLASH_WRPR[23：16])
0x1FFF F810~ 0x1FFF F9FF	**存储用户自定义数据**

每次系统复位后，选项字节装载器（OBL）读出信息块 0x1FFF F800~0x1FFF F80F 的数据，并保存在选项字节寄存器（ FLASH_OBR）中；每个选择位都在信息块中有它的反码位，在装载选择位时反码位用于验 证选择位是否正确，如果有任何的差别，将产生一个选项字节错误标志（OPTERR）。当发 生选项字节错误时，对应的选项字节被强置为 0xFF。当选项字节和它的反码均为 0xFF 时（擦除后的状态），则关闭上述验证功能。所有的选择位（不包括它们的反码位）用于配置该微控制器，CPU 可以读选项字节寄存器

3.4 Flash 寄存器描述

3.4.1 Flash 访问控制寄存器（FLASH_ACR）

地址偏移：0x00
复位值：0x0000 0030

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRFTBS	PRFTBE	HLFCYA	LATENCY[2:0]		
										r	rw	rw	rw	rw	rw

位 31:6	保留，必须保留复位值。
位 5	**PRFTBS**： 预取缓冲区状态该位提供预取缓冲区的状态 0：预取缓冲区被禁止 1：预取缓冲区被使能
位 4	**PRFTBS**： 预取缓冲区使能 0：禁止预取指； 1：使能预取指

位 3	**HLFCYA:**闪存半周期访问使能 (Flash half cycle access enable) 0：禁止半周期访问 1：启用半周期访问
位 2:0	**LATENCY[2:0]：** 等待周期；本位预设 HCLK 周期和 Flash 访问时间的比率关系； 000：零等待周期，适用于 0<HCLK≤24MHz； 001：1 个等待周期，适用于 24MHz<HCLK≤48MHz 010：2 个等待周期，适用于 48MHz<HCLK≤72MHz 011：3 个等待周期 100：7 个等待周期 101：9 个等待周期 110：19 个等待周期 111：39 个等待周期 LATENCY 配置为 011~111 适用于 CPU 可以在极低频率运行的应用程序，通过配置大的等待周期可以降低整个芯片的功耗。 通过设置更大的 wait-cycle 值，在某些无需高速处理能力的应用场景下 MCU 的功耗可以得到进一步的降低。

3.4.2 Flash 关键字寄存器（FLASH_KEYR）

地址偏移：0x04
复位值：xxxx xxxx
所有寄存器位全部只写，如果读之会返回 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FKEYR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FKEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 31:0	**FKEYR：**关键字 该位用于输入关键字以解锁 Flash

3.4.3 Flash 选项关键字寄存器 (FLASH_OPTKEYR)

地址偏移 ： 0x08
复位值 ： xxxx xxxx
所有寄存器位全部只写，如果读之会返回 0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPTKEYR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTKEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 31:0	**OPTKEYR：**选项字节关键字 该位用于输入关键字以解锁 OPTWRE

.

3.4.4 Flash 状态寄存器 (FLASH_SR)

地址偏移 ： 0x0C
复位值 ： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EOP	WRPRT ERR	Res.	PG ERR	Res.	BSY
										rw	rw		rw		r

位 31:6	保留，必须保留复位值。
位 5	**EOP:** 操作结束；当 Flash 操作（写/擦除）完成时由硬件置位。 软件写 1 后可清零。 *注意：只有成功的写或擦除操作才会由硬件置位 EOP。*
位 4	**WRPRTERR:** 写保护错误标志，当出现对写保护区域的写操作时被硬件置位。软件写 1 后可清零
位 3	保留，必须保留复位值。
位 2	**PGERR:** 写入错误标志 半字编程时，如果被编程区域的值不为 ’0xFFFF’ ，或者字编程时，如果被编程区域的值不为 ‘0xffff ffff’ ，则执行写入操作时被硬件置位。 软件写 1 后可清零
位 1	保留，必须保留复位值。
位 0	**BSY:** 忙标志 该位标明 Flash 操作处于过程中 当开始 Flash 操作的时候被硬件置位，当操作结束时或发生错误时被硬件清零。

3.4.5 Flash 控制寄存器 (FLASH_CR)

地址偏移：0x10
复位值：0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res	EOPIE	Res.	ERRIE	OPTWRE	Res.	LOCK	STRT	OPTER	OPT PG	Res.	MER	PER	PG
			rw		rw	rw		rw	rw	rw	rw		rw	rw	rw

位 31:13	保留，必须保留复位值。
位 12	**EOPIE:** 操作结束中断使能 该位使能操作结束中断，使能 FLASH_SR 中的 EOP 位变成 1 的时候产生中断请求 0: 中断禁止 1: 中断使能
位 11	保留，必须保留复位值

位 10	**ERRIE:** 错误中断使能 该位使能操作错误中断，使能 FLASH_SR 中的 PGERR/WRPRTERR 位变成 1 的时候产生中断请求。 0：中断禁止 1：中断使能
位 9	**OPTWRE:** 选项字节写使能 该位为 1 时，选项字节即允许改写。对 FLASH_OPTKEYR 寄存器写入正确的关键字序列就可以将它置 1。 该位可软件清零
位 8	保留，必须保留复位值
位 7	**LOCK:** 锁定 Flash 标志 只能写 1。 当该位为 1 时，表明 Flash 为锁定状态。 该位可以解锁时序来清零，当发现解锁不成功时，该位就一直为 1 了，除非下次复位重新操作
位 6	**STRT:** 启动 该位会触发一个擦除操作，仅由软件置 1，仅会在 BSY 被清零时清零
位 5	**OPTER:** 选项字节擦除 选项字节擦除时选择
位 4	**OPTPG:** 选项字节写入 选项字节写入时选择
位 3	保留，必须保留复位值。
位 2	**MER:** 整片擦除 整片擦除时选择
位 1	**PER:** 页擦除 页擦除时选择
位 0	**PG:** 半字写入 Flash 写入时选择

3.4.6 Flash 地址寄存器 (FLASH_AR)

地址偏移 ：0x14

复位值 ： 0x0000 0000

本寄存器由硬件根据当前和上次操作的地址更新。对于页擦除操作，该寄存器该由软件配置还指明要擦除的页或半页。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FAR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	6	5	4	3	2	1	0	
FAR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 31:0	**FAR：** Flash 地址 当 HPG 位被选中时，选择待擦除的半页的地址，或当 PER 位被选中时，选择待擦除的页地址

注意：　当 BSY 为 1 时，写被锁定

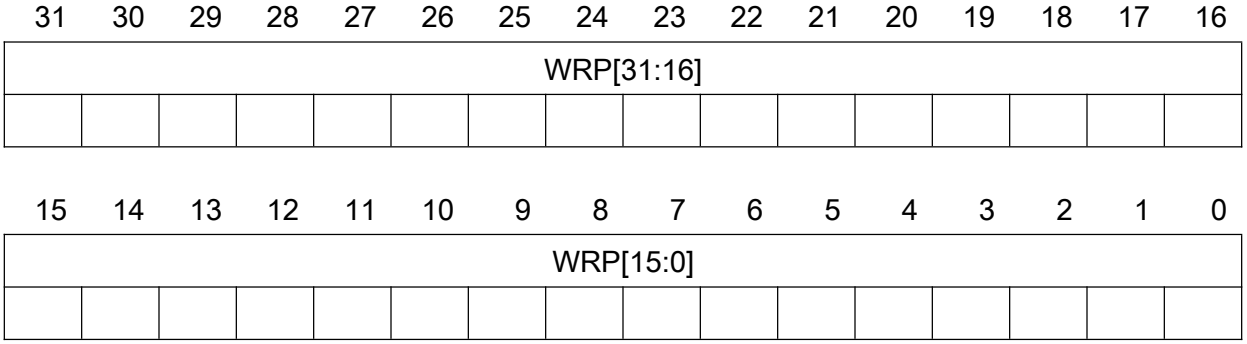
3.4.7 Flash 选项字节寄存器 (FLASH_OBR)

地址偏移 ： 0x1C
复位值 ： 0x03FF FFFC
本寄存器的复位值取决于选项字节的写入值，OPTERR 位的复位值取决于复位时选项字节加 载环节中比较选项字节及其补码的结果。

位 31:26	保留，必须保留复位值
位 25:18	Data1
位 17:10	Data0
位 9:5	保留，必须保留复位值
位 4	**nRST_STDBY:** 进入待机模式时的复位事件 0：当进入待机模式时产生复位 1：进入待机模式时不产生复位
位 3	**nRST_STOP:** 进入停机模式时的复位事件 0：当进入停机 (STOP) 模式时产生复位 1：进入停机 (STOP) 模式时不产生复位
位 2	**WDG_SW:** 选择看门狗事件 0：硬件看门狗 1：软件看门狗
位 1	**RDPRT:** 读保护 为 1 表示已经读保护了（只读）
位 0	**OPTERR:** 选项字节错误 当置位时，表明加载选项字节时反码校验错误。（只读）

3.4.8 Flash 写保护寄存器 (FLASH_WRPR)

地址偏移 ： 0x20
复位值 ： 0xFFFF FFFF
本寄存器的复位值取决于选项字节的写入值



位 31:0	**WRP**：写保护，保持由 OBL 载入的写保护选项字。（只读） 0：写保护激活 1：写保护禁止

3.4.9 Flash 控制寄存器 2 (FLASH_ECR)

地址偏移 ： 0x70

复位值 ： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES												WPG	RES		HPER
												R/W			R/W

位 31:4	保留，必须保留复位值
位 3	**WPG:** 字编程
位 2:1	保留，必须保留复位值
位 0	**HPER:** 半页擦除

注意： WPG，HPER，PG，PER，MER，OPTPG，OPTER 同一个时刻只能有一个位被置 1

3.4.10 寄存器映像

下表列出了寄存器映像和复位值

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	FLASH_ACR	Reserved																										PRFTBS	PRFTBE	HLFCYA	LATENCY [2:0]		
	Reset value																											1	1	0	0	0	0
0x004	FLASH_KEYR	FKEYR[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x008	FLASH_OPTKEYR	OPTKEYR[31:0]																															
	Reset Value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x00C	FLASH_SR	Reserved																										EOP	WRPRTERR	Reserved	PGERR	ERLYBSY	BSY
	Reset value																											0	0		0	0	0
0x010	FLASH_CR	Reserved																			EOPIE	Reserved	ERRIE	OPTWRE	Reserved	LOCK	STRT	OPTER	OPTPG	Reserved	MER	PER	PG
	Reset value																				0	0	0	0		1	0	0	0		0	0	0
0x014	FLASH_AR	FAR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	Reserved																																
0x01C	FLASH_OBR	Reserved							Data1							Data0								Not used				nRST_STDBY	nRST_STOP	WDG_SW	RDPRT		OPTERR
	Reset value								1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0		0
0x020	FLASH_WRPR	WRP[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x070	FLASH_ECR Reset value	保留																											WPG	保留			HPER

3.4.11 代码示例

512 字节擦除示例

如果要做擦除可以参考下面的代码，下面的例子以擦除 512bytes 为例：

```
FLASH_Unlock( ) ;
  *((volatile unsigned int *)(0x40022024)) = 0x00000001 ;
  FLASH->AR = 0xxxxx ;     // the page address
  FLASH->CR = 0x00000040 ;
  while( ( FLASH->SR & 0x00000001 ) == 0x00000001 ) ;
  FLASH->SR = 0xffffffff ;
```

字编程示例

```
FLASH_Unlock( ) ;
  *((volatile unsigned int *)(0x40022024)) = 0x00000008 ;
  *((volatile unsigned int *)(0xxxxx)) = 0xyyyyyyyy ;
//0xxxxx is the word-program aim address, 0xyyyyyyyy it the data to be
//programmed in
  while( ( FLASH->SR & 0x00000001 ) == 0x00000001 ){} ;
  FLASH->SR = 0xffffffff ;
```

4 CRC 计算单元（CRC）

4.1 CRC 简介

循环冗余校验(CRC)计算单元是根据固定的生成多项式得到任一 32 位全字的 CRC 计算结果。

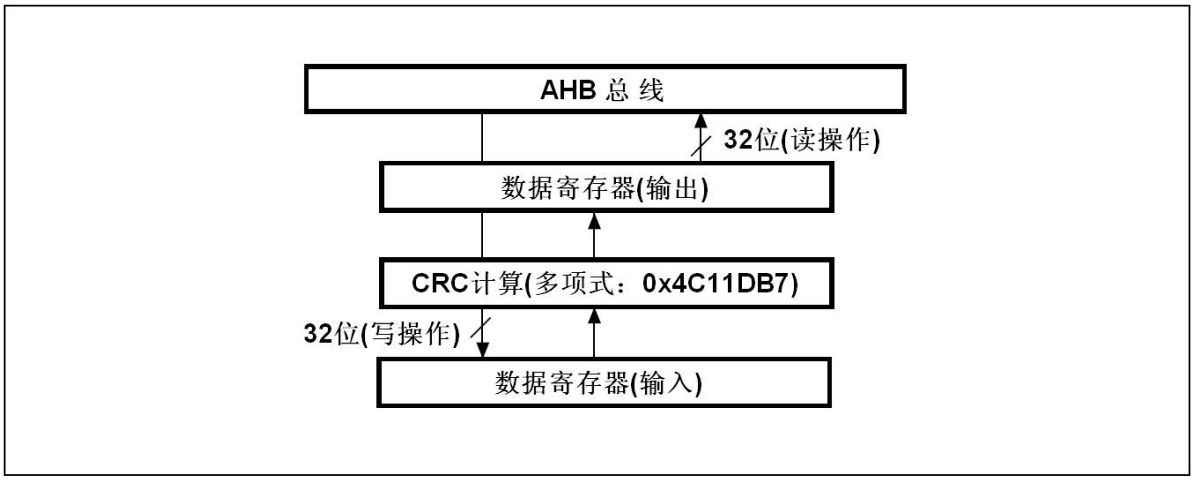
在其他的应用中，CRC 技术主要应用于核实数据传输或者数据存储的正确性和完整性。标准 EN/IEC 60335-1 即提供了一种核实闪存存储器完整性的方法。CRC 计算单元可以在程序运行时计算出软件的标识，之后与在连接时生成的参考标识比较，然后存放在指定的存储器空间。

4.2 CRC 主要特性

- 使用 CRC-32(以太网)多项式：0x4C11DB7
  - $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^4 + X^2 + X + 1$
- 一个 32 位数据寄存器用于输入 / 输出
- CRC 计算时间：4 个 AHB 时钟周期(HCLK)
- 通用 8 位寄存器(可用于存放临时数据)

下图为 CRC 计算单元框图

图 2 CRC 计算单元框图



4.3 CRC 功能描述

CRC 计算单元含有 1 个 32 位数据寄存器：

- 对该寄存器进行写操作时，作为输入寄存器，可以输入要进行 CRC 计算的新数据。
- 对该寄存器进行读操作时，返回上一次 CRC 计算的结果。

每一次写入数据寄存器，其计算结果是前一次 CRC 计算结果和新计算结果的组合(对整个 32 位字进行 CRC 计算，而不是逐字节地计算)。

在 CRC 计算期间会暂停 CPU 的写操作，因此可以对寄存器 CRC_DR 进行背靠背写入或者连续地 写-读操作。

可以通过设置寄存器 CRC_CR 的 RESET 位来重置寄存器 CRC_DR 为 0xFFFF FFFF。该操作不影响寄存器 CRC_IDR 内的数据。

4.4 CRC 寄存器

CRC 计算单元包括 2 个数据寄存器和 1 个控制寄存器。

4.4.1 数据寄存器(CRC_DR)

地址偏移：0x00

复位值：0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31: 0	数据寄存器位 写入 CRC 计算器的新数据时，作为输入寄存器 读取时返回 CRC 计算的结果

4.4.2 独立数据寄存器(CRC_IDR)

地址偏移：0x04

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								IDR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

位 31:8	保留。
位 7:0	通用 8 位数据寄存器位 可用于临时存放 1 字节的数据。 寄存器 CRC_CR 的 RESET 位产生的 CRC 复位对本寄存器没有影响

译注：此寄存器不参与 CRC 计算，可以存放任何数据。

4.4.3 控制寄存器(CRC_CR)

地址偏移：0x08

复位值：0x0000 0000

w

位 31:1	保留。
位 0	RESET 位 复位 CRC 计算单元，设置数据寄存器为 0xFFFF FFFF。 只能对该位写'1'，它由硬件自动清'0'。

4.4.4 寄存器映像

下表列出了 CRC 的寄存器映像和复位值

表 4 CRC 计算单元寄存器映像和复位值

偏移	寄存器	31~24	23~16	15~8	7	6	5	4	3	2	1	0
0x00	CRC_DR	数据寄存器 0xFFFF FFFF										
	复位值											
0x04	CRC_IDR	保留			独立的数据寄存器 0x00							
	复位值											
0x08	CRC_CR	保留								保留 0		RESET0
	复位值											

关于寄存器的起始地址，参见表 1。

5 电源控制（PWR）

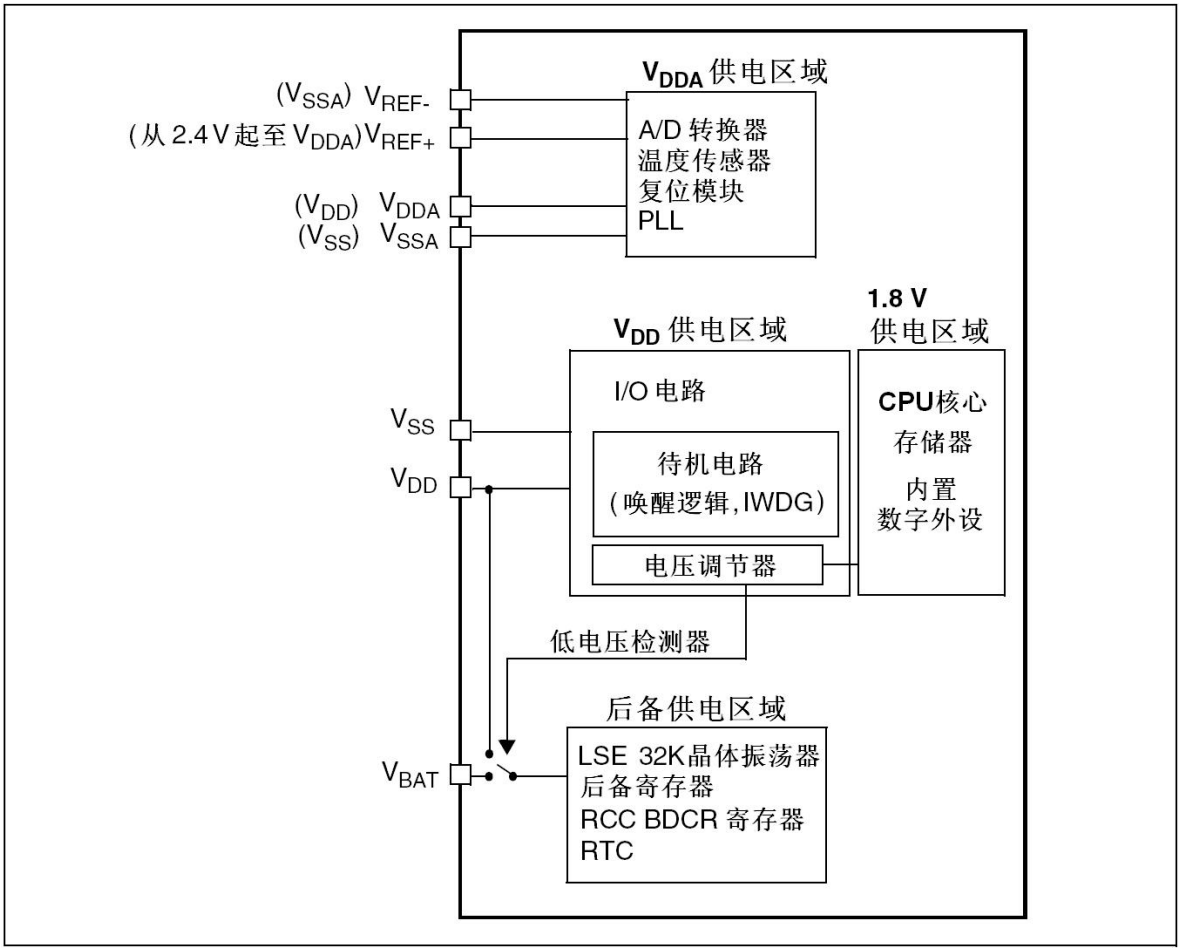
5.1 电源

HK32F103 的工作电压(V_{DD})为 1.8～3.6V。通过内置的电压调节器提供所需的 1.8V 电源。 当主电源 V_{DD} 掉电后，通过 V_{BAT} 脚为实时时钟(RTC)和备份寄存器提供电源。

HKF103片内数字逻辑的电源由片内LDO提供。

片内LDO的输出电压可通过寄存器设置，以方便软件根据应用场景最大程度的优化芯片的电流功耗。芯片RUN模式和STOP模式的LDO输出电压可以分别独立设置。

图 3 电源框图



注： *VDDA 和 VSSA 必须分别联到 VDD 和 VSS。*

5.1.1 独立的 A/D 转换器供电和参考电压

为了提高转换的精确度，ADC 使用一个独立的电源供电，过滤和屏蔽来自印刷电路板上的毛刺干扰。

- ADC 的电源引脚为 V_{DDA}
- 独立的电源地 V_{SSA}

如果有 V_{REF-}引脚(根据封装而定)，它必须连接到 V_{SSA}。

没有 V_{REF+}和 V_{REF-}引脚，他们在芯片内部与 ADC 的电源(V_{DDA})和地(V_{SSA})相联。

5.1.2 电池备份区域

使用电池或其他电源连接到 V_{BAT} 脚上，当 V_{DD} 断电时，可以保存备份寄存器的内容和维持 RTC 的功能。

V_{BAT} 脚为 RTC、LSE 振荡器和 PC13 至 PC15 端口供电，可以保证当主电源被切断时 RTC 能继续工作。切换到 V_{BAT} 供电的开关，由复位模块中的掉电复位功能控制。

警告：

在 V_{DD} 上升阶段($t_{RSTTEMPO}$)或者探测到 PDR(掉电复位)之后，V_{BAT} 和 V_{DD} 之间的电源开关仍会保持连接在 V_{BAT}。

在 V_{DD} 上升阶段，如果 V_{DD} 在小于 $t_{RSTTEMPO}$ 的时间内达到稳定状态(关于 $t_{RSTTEMPO}$ 数值可参考数据手册中的相关部分)，且 $V_{DD} > V_{BAT} + 0.6V$ 时，电流可能通过 V_{DD} 和 V_{BAT} 之间的内部二极管注入到 V_{BAT}。

如果与 V_{BAT} 连接的电源或者电池不能承受这样的注入电流，强烈建议在外部 V_{BAT} 和电源之间连接一个低压降二极管。

如果在应用中没有外部电池，建议 V_{BAT} 在外部连接到 V_{DD} 并连接一个 100nF 的陶瓷滤波电容。

当备份区域由 V_{DD}(内部模拟开关连到 V_{DD})供电时，下述功能可用：

- PC14 和 PC15 可以用于 GPIO 或 LSE 引脚
- PC13 可以作为通用 I/O 口、TAMPER 引脚、RTC 校准时钟、RTC 闹钟或秒输出(参见第 7 章：备份寄存器(BKP))

注：因为模拟开关只能通过少量的电流(3mA)，在输出模式下使用 PC13 至 PC15 的 I/O 口功能是有限制的：速度必须限制在 2MHz 以下，最大负载为 30pF，而且这些 I/O 口绝对不能当作电流源(如驱动 LED)。

当后备区域由 V_{BAT} 供电时(VDD 消失后模拟开关连到 V_{BAT})，可以使用下述功能：

- PC14 和 PC15 只能用于 LSE 引脚
- PC13 可以作为 TAMPER 引脚、RTC 闹钟或秒输出(参见第 7.4.2 节：RTC 时钟校准寄存器(BKP_RTCCR))

5.1.3 电压调节器

复位后调节器总是使能的。根据应用方式它以 3 种不同的模式工作。

- 运转模式：调节器以正常功耗模式提供 1.8V 电源(内核，内存和外设)。
- 停止模式：调节器以低功耗模式提供 1.8V 电源，以保存寄存器和 SRAM 的内容。
- 待机模式：调节器停止供电。除了备用电路和备份域外，寄存器和 SRAM 的内容全部丢失。

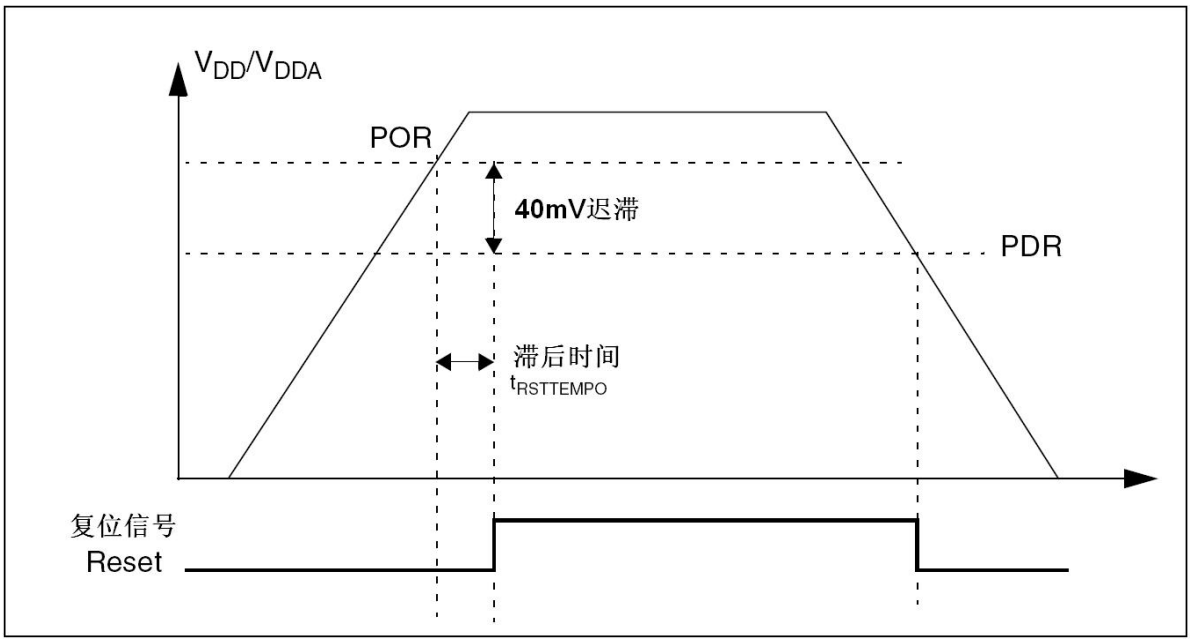
5.2 电源管理器

5.2.1 上电复位(POR)和掉电复位(PDR)

HK32 内部有一个完整的上电复位(POR)和掉电复位(PDR)电路，当供电电压达到 2V 时系统既能正常工作。

当 V_{DD}/V_{DDA} 低于指定的限位电压 V_{POR}/V_{PDR} 时，系统保持为复位状态，而无需外部复位电路。关于上电复位和掉电复位的细节请参考数据手册的电气特性部分。

图 4 上电复位和掉电复位的波形图



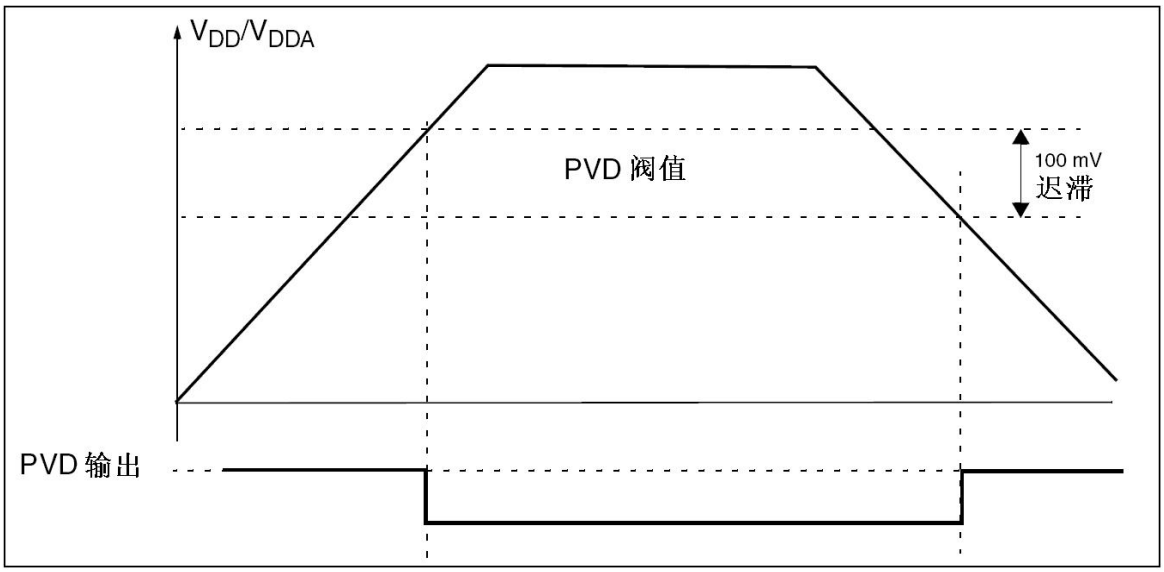
5.2.2 可编程电压监测器(PVD)

用户可以利用 PVD 对 V_{DD} 电压与电源控制寄存器(PWR_CR)中的 PLS[2:0]位进行比较来监控电源，这几位选择监控电压的阀值。

通过设置 PVDE 位来使能 PVD。

电源控制/状态寄存器(PWR_CSR)中的 PVDO 标志用来表明 V_{DD} 是高于还是低于 PVD 的电压阀值。该事件在内部连接到外部中断的第 16 线，如果该中断在外部中断寄存器中是使能的，该事件就会产生中断。当 V_{DD} 下降到 PVD 阀值以下和(或)当 V_{DD} 上升到 PVD 阀值之上时，根据外部中断第 16 线的上升/下降边沿触发设置，就会产生 PVD 中断。例如，这一特性可用于用于执行紧急关闭任务。

图 5 PVD 的门限



5.3 低功耗模式

在系统或电源复位以后，微控制器处于运行状态。当 CPU 不需继续运行时，可以利用多种低功 耗模式来节省功耗，例如等待某个外部事件时。用户需要根据最低电源消耗、最快速启动时间和可用的唤醒源等条件，选定一个最佳的低功耗模式。

HK32F10xxx 有三种低功耗模式：

- 睡眠模式(Cortex™-M3 内核停止，所有外设包括 Cortex-M3 核心的外设，如 NVIC、系统时 钟(SysTick)等仍在运行)
- 停止模式(所有的时钟都已停止)
- 待机模式(1.8V 电源关闭)

此外，在运行模式下，可以通过以下方式中的一种降低功耗：

- 降低系统时钟
- 关闭 APB 和 AHB 总线上未被使用的外设时钟。

表 5 低功耗模式一览

模式	进入	唤醒	对 1.8V 区域 时钟的	对 V_{DD} 区域时钟的影响	电压调节器
睡眠 (SLEEP-NOW 或 SLEEP-ON-EXIT)	WFI	任一中断	CPU 时钟关，对其他时钟和 ADC 时钟无影	无	开
	WFE	唤醒事件			
停机	PDDS 和 LPDS 位 +SLEEPDEEP 位 +WFI 或 WFE	任一外部中断(在外部中断寄存器中设置)	关闭所有 1.8V 区域的时钟	HSI 和 HSE 的振荡器关闭	开启或处于低功耗模式(依据电源控制寄存器(PWR_CR)的设
待机	PDDS 位 +SLEEPDEEP 位 +WFI 或 WFE	WKUP 引脚的上升沿、RTC 闹钟事件、NRST 引脚上的外部复位、IWDG 复位			关

5.3.1 降低系统时钟

在运行模式下，通过对预分频寄存器进行编程，可以降低任意一个系统时钟(SYSCLK、HCLK、PCLK1、PCLK2)的速度。进入睡眠模式前，也可以利用预分频器来降低外设的时钟。 详见第 8.3.2 节：时钟配置寄存器(RCC_CFGR)。

5.3.2 外部时钟的控制

在运行模式下，任何时候都可以通过停止为外设和内存提供时钟(HCLK 和 PCLKx)来减少功耗。为了在睡眠模式下更多地减少功耗，可在执行 WFI 或 WFE 指令前关闭所有外设的时钟。

通过设置 AHB 外设时钟使能寄存器(RCC_AHBENR)、APB2 外设时钟使能寄存器(RCC_APB2ENR)和 APB1 外设时钟使能寄存器(RCC_APB1ENR)来开关各个外设模块的时钟。

5.3.3 睡眠模式

进入睡眠模式

通过执行 WFI 或 WFE 指令进入睡眠状态。根据 Cortex™-M3 系统控制寄存器中的 SLEEPONEXIT 位的值，有两种选项可用于选择睡眠模式进入机制：

- SLEEP-NOW：如果 SLEEPONEXIT 位被清除，当 WRI 或 WFE 被执行时，微控制器立即进入睡眠模式。
- SLEEP-ON-EXIT：如果 SLEEPONEXIT 位被置位，系统从最低优先级的中断处理程序中退 出时，微控制器就立即进入睡眠模式。

在睡眠模式下，所有的 I/O 引脚都保持它们在运行模式时的状态。

关于如何进入睡眠模式，更多的细节参考表 6 和表 7。

退出睡眠模式

如果执行 WFI 指令进入睡眠模式，任意一个被嵌套向量中断控制器响应的外设中断都能将系统从睡眠模式唤醒。

如果执行 WFE 指令进入睡眠模式，则一旦发生唤醒事件时，微处理器都将从睡眠模式退出。唤醒事件可以通过下述方式产生：

- 在外设控制寄存器中使能一个中断，而不是在 NVIC(嵌套向量中断控制器)中使能，并且在 Cortex-M3 系统控制寄存器中使能 SEVONPEND 位。当 MCU 从 WFE 中唤醒后，外设的中断挂起位和外设的 NVIC 中断通道挂起位(在 NVIC 中断清除挂起寄存器中)必须被清除。
- 配置一个外部或内部的 EXIT 线为事件模式。当 MCU 从 WFE 中唤醒后，因为与事件线对应的挂起位未被设置，不必清除外设的中断挂起位或外设的 NVIC 中断通道挂起位。

该模式唤醒所需的时间最短，因为没有时间损失在中断的进入或退出上。 关于如何退出睡眠模式，更多的细节参考表 9 和表 10。

表 6 SLEEP-NOW 模式

SLEEP-NOW 模式	**说明**
进入	在以下条件下执行 WFI(等待中断)或 WFE(等待事件)指令： – SLEEPDEEP = 0 和 – SLEEPONEXIT = 0 参考 Cortex-M3 系统控制寄存器。
退出	如果执行 WFI 进入睡眠模式： 中断：参考中断向量表(表 37) 如果执行 WFE 进入睡眠模式： 唤醒事件：参考唤醒事件管理(第 10.2.3 节)
唤醒延时	无

表 7 SLEEP-ON-EXIT 模式

SLEEP-ON_EXIT 模式	**说明**
进入	在以下条件下执行 WFI 指令： – SLEEPDEEP = 0 和 – SLEEPONEXIT = 1 参考 Cortex™-M3 系统控制寄存器
退出	中断：参考中断向量表(表 37)
唤醒延时	无

5.3.4 停止模式

停止模式是在 Cortex™-M3 的深睡眠模式基础上结合了外设的时钟控制机制，在停止模式下电压调节器可运行在正常或低功耗模式。此时在 1.8V 供电区域的的所有时钟都被停止，PLL、HSI 和HSE RC 振荡器的功能被禁止，SRAM 和寄存器内容被保留下来。

在停止模式下，所有的 I/O 引脚都保持它们在运行模式时的状态。

进入停止模式

关于如何进入停止模式，详见表 8。

在停止模式下，通过设置电源控制寄存器(PWR_CR)的 LPDS 位使内部调节器进入低功耗模式，能够降低更多的功耗。

如果正在进行闪存编程，直到对内存访问完成，系统才进入停止模式。如果正在进行对 APB 的访问，直到对 APB 访问完成，系统才进入停止模式。可以通过对独立的控制位进行编程，可选择以下功能：

- 独立看门狗(IWDG)：可通过写入看门狗的键寄存器或硬件选择来启动 IWDG。一旦启动了独立看门狗，除了系统复位，它不能再被停止。详见 16.3 节。
- 实时时钟(RTC)：通过备份域控制寄存器(RCC_BDCR)的 RTCEN 位来设置。
- 内部 RC 振荡器(LSI RC)：通过控制/状态寄存器(RCC_CSR)的 LSION 位来设置。
- 外部 32.768kHz 振荡器(LSE)：通过备份域控制寄存器 (RCC_BDCR)的 LSEON 位设置。

在停止模式下，如果在进入该模式前 ADC 和 DAC 没有被关闭，那么这些外设仍然消耗电流。通过设置寄存器 ADC_CR2 的 ADON 位和寄存器 DAC_CR 的 ENx 位为 0 可关闭这 2 个外设。

退出停止模式

关于如何退出停止模式，详见下表。 当一个中断或唤醒事件导致退出停止模式时，

HSI RC 振荡器被选为系统时钟。

当电压调节器处于低功耗模式下，当系统从停止模式退出时，将会有一段额外的启动延时。如果在停止模式期间保持内部调节器开启，则退出启动时间会缩短，但相应的功耗会增加。

表 8 停止模式

停止模式	说明
进入	在以下条件下执行 WFI(等待中断)或 WFE(等待事件)指令： – 设置 Cortex-M3 系统控制寄存器中的 SLEEPDEEP 位 – 清除电源控制寄存器(PWR_CR)中的 PDDS 位 – 通过设置 PWR_CR 中 LPDS 位选择电压调节器的模式 注：为了进入停止模式，所有的外部中断的请求位(挂起寄存器(EXTI_PR))和 RTC 的闹钟标志都必须被清除，否则停止模式的进入流程将会被跳过，程序继续运行。
退出	如果执行 WFI 进入停止模式： 设置任一外部中断线为中断模式(在 NVIC 中必须使能相应的外部中断向量)。参见中断向量表(表 37)。 如果执行 WFE 进入停止模式： 设置任一外部中断线为事件模式。参见唤醒事件管理(第 10.2.3 节)。
唤醒延时	HSI RC 唤醒时间 + 电压调节器从低功耗唤醒的时间。

5.3.5 待机模式

待机模式可实现系统的最低功耗。该模式是在 Cortex-M3 深睡眠模式时关闭电压调节器。整个1.8V 供电区域被断电。PLL、HSI 和 HSE 振荡器也被断电。SRAM 和寄存器内容丢失。只有备份的寄存器和待机电路维持供电(见图 3)。

进入待机模式

关于如何进入待机模式，详见表 9。 可以通过设置独立的控制位，选择以下待机模式的功能：

- 独立看门狗(IWDG)：可通过写入看门狗的键寄存器或硬件选择来启动 IWDG。一旦启动了独立看门狗，除了系统复位，它不能再被停止。详见 16.3 节。
- 实时时钟(RTC)：通过备用区域控制寄存器(RCC_BDCR)的 RTCEN 位来设置。
- 内部 RC 振荡器(LSIRC)：通过控制/状态寄存器(RCC_CSR)的 LSION 位来设置。
- 外部 32.768kHz 振荡器(LSE)：通过备用区域控制寄存器(RCC_BDCR)的 LSEON 位设置。

退出待机模式

当一个外部复位(NRST 引脚)、IWDG 复位、WKUP 引脚上的上升沿或 RTC 闹钟事件的上升沿发生时(见图 131：简化的 RTC 框图)，微控制器从待机模式退出。从待机唤醒后，除了电源控制/状态寄存器(PWR_CSR)(见第 6.4.2 节)，所有寄存器被复位。

从待机模式唤醒后的代码执行等同于复位后的执行(采样启动模式引脚、读取复位向量等)。电源控制/状态寄存器(PWR_CSR)(见第 6.4.2 节)将会指示内核由待机状态退出。

关于如何退出待机模式，详见下表。

表 9 待机模式

待机模式	说明
进入	在以下条件下执行 WFI(等待中断)或 WFE(等待事件)指令： - 设置 Cortex™-M3 系统控制寄存器中的 SLEEPDEEP 位 - 设置电源控制寄存器(PWR_CR)中的 PDDS 位 - 清除电源控制/状态寄存器(PWR_CSR)中的 WUF 位
退出	WKUP 引脚的上升沿、RTC 闹钟事件的上升沿、NRST 引脚上外部复位、IWDG 复位。
唤醒延时	复位阶段时电压调节器的启动。

待机模式下的输入/输出端口状态

在待机模式下，所有的 I/O 引脚处于高阻态，除了以下的引脚：

- 复位引脚(始终有效)
- 当被设置为防侵入或校准输出时的 TAMPER 引脚
- 被使能的唤醒引脚

调试模式

默认情况下，如果在进行调试微处理器时，使微处理器进入停止或待机模式，将失去调试连接。这是因为 Cortex™-M3 的内核失去了时钟。

然而，通过设置 DBGMCU_CR 寄存器中的某些配置位，可以在使用低功耗模式下调试软件。更多的细节请参考第 25.15.1 节：低功耗模式的调试支持。

5.3.6 低功耗模式下的自动唤醒(AWU)

RTC 可以在不需要依赖外部中断的情况下唤醒低功耗模式下的微控制器(自动唤醒模式)。RTC 提供一个可编程的时间基数，用于周期性从停止或待机模式下唤醒。通过对备份区域控制寄存器(RCC_BDCR)的 RTCSEL[1:0]位的编程，三个 RTC 时钟源中的二个时钟源可以选作实现此功能。

- 低功耗 32.768kHz 外部晶振(LSE) 该时钟源提供了一个低功耗且精确的时间基准。(在典型情形下消耗小于 1μA)
- 低功耗内部 RC 振荡器(LSI RC)
  使用该时钟源，节省了一个 32.768kHz 晶振的成本。但是 RC 振荡器将少许增加电源消耗。

为了用 RTC 闹钟事件将系统从停止模式下唤醒，必须进行如下操作：

- 配置外部中断线 17 为上升沿触发。
- 配置 RTC 使其可产生 RTC 闹钟事件。

如果要从待机模式中唤醒，不必配置外部中断线 17。

5.4 电源控制寄存器

可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

5.4.1 电源控制寄存器(PWR_CR)

地址偏移：0x00

复位值：0x0000 0000 (从待机模式唤醒时清除)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留							DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
							rw	rw	rw	rw	rw	rc_w1	rc_w1	rw	rw

位31:9	保留。始终读为 0。
位8	**DBP**：取消后备区域的写保护 在复位后，RTC 和后备寄存器处于被保护状态以防意外写入。设置这位允许写入这些寄存器。 0：禁止写入 RTC 和后备寄存器 1：允许写入 RTC 和后备寄存器 注：如果 RTC 的时钟是 HSE/128，该位必须保持为'1'。
位7:5	**PLS[2:0]**：PVD 电平选择 这些位用于选择电源电压监测器的电压阀值 000：2.2V　　100：2.6V 001：2.3V　　101：2.7V 010：2.4V　　110：2.8V 011：2.5V　　111：2.9V 注：详细说明参见数据手册中的电气特性部分。
位4	**PVDE**：电源电压监测器(PVD)使能 0：禁止 PVD 1：开启 PVD
位3	**CSBF**：清除待机位 始终读出为 0 0：无功效 1：清除 SBF 待机位(写)
位2	**CWUF**：清除唤醒位 始终读出为 0 0：无功效 1：2 个系统时钟周期后清除 WUF 唤醒位(写)
位1	**PDDS**：掉电深睡眠 与 LPDS 位协同操作 0：当 CPU 进入深睡眠时进入停机模式，调压器的状态由 LPDS 位控制。 1：CPU 进入深睡眠时进入待机模式。
位0	**LPDS**：深睡眠下的低功耗 PDDS=0 时，与 PDDS 位协同操作 0：在停机模式下电压调压器开启 1：在停机模式下电压调压器处于低功耗模式

5.4.2 电源控制/状态寄存器(PWR_CSR)

地址偏移：0x04

复位值：0x0000 0000 (从待机模式唤醒时不被清除)

与标准的 APB 读相比，读此寄存器需要额外的 APB 周期

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留							EWUP	保留					PVDO	SBF	WUF
							rw						r	r	r

位 31:9	保留。始终读为 0。
位8	**EWUP**：使能 WKUP 引脚 0：WKUP 引脚为通用 I/O。WKUP 引脚上的事件不能将 CPU 从待机模式唤醒 1：WKUP 引脚用于将 CPU 从待机模式唤醒，WKUP 引脚被强置为输入下拉的配置(WKUP 引脚上的上升沿将系统从待机模式唤醒) 注：在系统复位时清除这一位。
位7:3	保留。始终读为 0。
位2	**PVDO**：PVD 输出 当 PVD 被 PVDE 位使能后该位才有效 0：VDD/VDDA 高于由 PLS[2:0]选定的 PVD 阀值 1：V_{DD}/V_{DDA} 低于由 PLS[2:0]选定的 PVD 阀值 注：在待机模式下 PVD 被停止。因此，待机模式后或复位后，直到设置 PVDE 位之前，该位为0。
位1	**SBF**：待机标志 该位由硬件设置，并只能由 POR/PDR(上电/掉电复位)或设置电源控制寄存器(PWR_CR)的 CSBF 位清除。 0：系统不在待机模式 1：系统进入待机模式
位0	**WUF**：唤醒标志 该位由硬件设置，并只能由 POR/PDR(上电/掉电复位)或设置电源控制寄存器(PWR_CR)的 CWUF 位清除。 0：没有发生唤醒事件 1：在 WKUP 引脚上发生唤醒事件或出现 RTC 闹钟事件。 注：当 WKUP 引脚已经是高电平时，在(通过设置 EWUP 位)使能 WKUP 引脚时，会检测到一个额外的事件。

5.4.3 电源控制/状态寄存器(PWR_LDO)

地址偏移：0x10

复位值：0x0000 2538(复位时恢复默认值)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												LDO_VOUT			
												rw			

位 31:4	保留。
位3:0	**LDO_VOUT:** 设置 MCU 的 RUN 模式下，片内 LDO 的输出电压。 0000 : 1.20V 0001 : 1.24V 0010 : 1.28V 0011 : 1.32V 0100 : 1.36 V 0101 : 1.40 V 0110 : 1.44 V 0111 : 1.52 V 1000 : 1.56 V 1001 : 1.60 V 1010 : 1.64 V 1011 : 1.68V Others : 禁止设置。

注意： *PWR_LDO[31:4] ：**含有芯片内部测试寄存器；禁止改变其初始值，否则芯片功能可能异常！***

5.4.4 电源控制/状态寄存器(PWR_LDO_STOP)

地址偏移：0x14

复位值：0x0000 20b8(复位时恢复默认值)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												LDO_VOUTS			
												rw			

位 31:4	保留。
位3:0	**LDO_VOUTS:** 设置 MCU 的 STOP 模式下，片内 LDO 的输出电压。 0000 : 1.20V 0001 : 1.24V 0010 : 1.28V 0011 : 1.32V 0100 : 1.36 V 0101 : 1.40 V 0110 : 1.44 V 0111 : 1.52 V 1000 : 1.56 V 1001 : 1.60 V 1010 : 1.64 V 1011 : 1.68V Others : 禁止设置。

注意： *PWR_LDO_STOP[31:4]* ***：含有芯片内部测试寄存器；禁止改变其初始值，否则芯片功能可能异常！***

5.4.5 PWR 寄存器地址映像

以下表格列出所有 PWR 寄存器。

表 10 PWR 寄存器地址映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	PWR_CR	保留																							DBP	PLS [2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
	复位值																								0	0	0	0	0	0	0	0	0
004h	PWR_CSR	保留																							EWUP	保留					PVDO	SBF	WUF
	复位值																								0						0	0	0
010h	PWR_LDO	保留																													PWR_LDO [3:0]			
	复位值																														0	0	0	0
014h	PWR_LDOS	保留																													PWR_LDOS [3:0]			
	复位值																														0	0	0	0

关于寄存器的起始地址，参见表 1。

6 备份寄存器（BKP）

6.1 BKP 简介

备份寄存器是 42 个 16 位的寄存器，可用来存储 84 个字节的用户应用程序数据。他们处在备份域里，当 V_{DD} 电源被切断，他们仍然由 V_{BAT} 维持供电。当系统在待机模式下被唤醒，或系统复位或电源复位时，他们也不会被复位。

此外，BKP 控制寄存器用来管理侵入检测和 RTC 校准功能。

复位后，对备份寄存器和 RTC 的访问被禁止，并且备份域被保护以防止可能存在的意外的写操作。执行以下操作可以使能对备份寄存器和 RTC 的访问。

- 通过设置寄存器 RCC_APB1ENR 的 PWREN 和 BKPEN 位来打开电源和后备接口的时钟
- 电源控制寄存器(PWR_CR)的 DBP 位来使能对后备寄存器和 RTC 的访问。

6.2 BKP 特性

- 20 字节数据后备寄存器
- 用来管理防侵入检测并具有中断功能的状态/控制寄存器
- 用来存储 RTC 校验值的校验寄存器。
- 在 PC13 引脚(当该引脚不用于侵入检测时)上输出 RTC 校准时钟，RTC 闹钟脉冲或者秒脉冲

6.3 BKP 功能描述

6.3.1 侵入检测

当 TAMPER 引脚上的信号从'0'变成'1'或者从'1'变成'0'(取决于备份控制寄存器 BKP_CR 的 TPAL 位)，会产生一个侵入检测事件。侵入检测事件将所有数据备份寄存器内容清除。

然而为了避免丢失侵入事件，侵入检测信号是边沿检测的信号与侵入检测允许位的逻辑与，从而在侵入检测引脚被允许前发生的侵入事件也可以被检测到。

- **当 TPAL=0 时：**如果在启动侵入检测 TAMPER 引脚前(通过设置 TPE 位)该引脚已经为高电平，一旦启动侵入检测功能，则会产生一个额外的侵入事件(尽管在 TPE 位置'1'后并没有出 现上升沿)。
- **当 TPAL=1 时：**如果在启动侵入检测引脚 TAMPER 前(通过设置 TPE 位)该引脚已经为低电平，一旦启动侵入检测功能，则会产生一个额外的侵入事件(尽管在 TPE 位置'1'后并没有出 现下降沿)。

设置 BKP_CSR 寄存器的 TPIE 位为'1'，当检测到侵入事件时就会产生一个中断。

在一个侵入事件被检测到并被清除后，侵入检测引脚 TAMPER 应该被禁止。然后，在再次写入备份数据寄存器前重新用 TPE 位启动侵入检测功能。这样，可以阻止软件在侵入检测引脚上仍然有侵入事件时对备份数据寄存器进行写操作。这相当于对侵入引脚 TAMPER 进行电平检测。

注：　当 V_{DD} 电源断开时，侵入检测功能仍然有效。为了避免不必要的复位数据备份寄存器，TAMPER 引脚应该在片外连接到正确的电平。

6.3.2 RTC 校准

为方便测量，RTC 时钟可以经 64 分频输出到侵入检测引脚 TAMPER 上。通过设置 RTC 校验寄存 器(BKP_RTCCR)的 CCO 位来开启这一功能。

通过配置 CAL[6:0]位，此时钟可以最多减慢 121ppm。

6.4 BKP 寄存器描述

关于在寄存器描述里面所用到的缩写，可参考 2.1 节。可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

6.4.1 备份数据寄存器 x(BKP_DRx) (x = 1 … 10)

地址偏移：0x04 到0x28，0x40 到0xBC

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	9	5	4	3	2	1	0
D[15:0]															
rw															

位 15:0	**D[15:0]**：备份数据 这些位可以被用来写入用户数据。 注意：BKP_DRx 寄存器不会被系统复位、电源复位、从待机模式唤醒所复位。 它们可以由备份域复位来复位或(如果侵入检测引脚 TAMPER 功能被开启时)由侵入引脚事件复位。

6.4.2 RTC 时钟校准寄存器(BKP_RTCCR)

地址偏移：0x2C

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	9	5	4	3	2	1	0
保留						ASOS	ASOE	CCO	CAL[6:0]						
						rw	rw	rw	rw						

15:8	保留，始终读为 0。
位 9	**ASOS**：闹钟或秒输出选择(Alarm or second output selection) 当设置了 ASOE 位，ASOS 位可用于选择在 TAMPER 引脚上输出的是 RTC 秒脉冲还是闹钟脉信号。 0：输出 RTC 闹钟脉冲 1：输出秒脉冲 注：该位只能被后备区的复位所清除
位 8	**ASOE**：允许输出闹钟或秒脉冲(Alarm or second output enable) 根据 ASOS 位的设置，该位允许 RTC 闹钟或秒脉冲输出到 TAMPER 引脚上。 输出脉冲的宽度为一个 RTC 时钟的周期。设置了 ASOE 位时不能开启 TAMPER 的功能。 注：该位只能被后备区的复位所清除
位 7	**CCO**：校准时钟输出(Calibration clock output) 0：无影响 1：此位置 1 可以在侵入检测引脚输出经 64 分频后的 RTC 时钟。当 CCO 位置 1 时，必须关闭侵检测功能以避免检测到无用的侵入信号。 注：当 VDD 供电断开时，该位被清除。
位 6:0	**CAL[6:0]**：校准值(Calibration value) 校准值表示在每 2^{20} 个时钟脉冲内将有多少个时钟脉冲被跳过。这可以用来对 RTC 进行校准，以 1000000/2^{20}ppm 的比例减慢时钟。 RTC 时钟可以被减慢 0~121ppm。

6.4.3 备份控制寄存器(BKP_CR)

偏移地址：0x30

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														TPAL	TPE
														rw	rw

位 15:2	保留，始终读为 0。
位 1	**TPAL**：侵入检测 TAMPER 引脚有效电平(TAMPER pin active level) 0：侵入检测 TAMPER 引脚上的高电平会清除所有数据备份寄存器(如果 TPE 位为 1) 1：侵入检测 TAMPER 引脚上的低电平会清除所有数据备份寄存器(如果 TPE 位为 1)
位 0	**TPE**：启动侵入检测 TAMPER 引脚(TAMPER pin enable) 0：侵入检测 TAMPER 引脚作为通用 IO 口使用 1：开启侵入检测引脚作为侵入检测使用

注：同时设置 TPAL 和 TPE 位总是安全的。然而，同时清除两者会产生一个假的侵入事件。因此，推荐只在 TPE 为 0 时才改变 TPAL 位的状态。

6.4.4 备份控制/状态寄存器(BKP_CSR)

偏移地址：0x34

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						TIF	TEF	保留					TPIE	CTI	CTE
						r	r						rw	rw	rw

位 15:10	保留，始终读为 0。
位 9	**TIF**：侵入中断标志(Tamper interrupt flag) 当检测到有侵入事件且 TPIE 位为 1 时，此位由硬件置 1。通过向 CTI 位写 1 来清除此标志位(同也清除了中断)。如果 TPIE 位被清除，则此位也会被清除。 0：无侵入中断 1：产生侵入中断 注意：仅当系统复位或由待机模式唤醒后才复位该位
位 8	**TEF**：侵入事件标志(Tamper event flag) 当检测到侵入事件时此位由硬件置 1。通过向 CTE 位写 1 可清除此标志位 0：无侵入事件 1：检测到侵入事件 注：侵入事件会复位所有的 BKP_DRx 寄存器。只要 TEF 为 1，所有的 BKP_DRx 寄存器就一直持复位状态。当此位被置 1 时，若对 BKP_DRx 进行写操作，写入的值不会被保存。
位 7:3	保留，始终读为 0。
位 2	**TPIE**：允许侵入 TAMPER 引脚中断(TAMPER pin interrupt enable) 0：禁止侵入检测中断 1：允许侵入检测中断(BKP_CR 寄存器的 TPE 位也必须被置 1) 注 1：侵入中断无法将系统内核从低功耗模式唤醒。 注 2：仅当系统复位或由待机模式唤醒后才复位该位。

位 1	**CTI**：清除侵入检测中断(Clear tamper interrupt) 此位只能写入，读出值为 0。 0：无效 1：清除侵入检测中断和 TIF 侵入检测中断标志
位 0	**CTE**：清除侵入检测事件(Clear tamper event) 此位只能写入，读出值为 0。 0：无效 1：清除 TEF 侵入检测事件标志(并复位侵入检测器)。

6.4.5 控制寄存器 (BKP_LSE_CTL)

偏移地址：0x3C
复位值：0x 000082c2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AGC_EN	保留					SEL_AGC		AUTO_IOP	NFBYP	保留	IOP_MON		保留	IOP	
rw						rw		rw	rw		r			rw	

位 31：16	保留
位 15	**AGC_EN**： 0：关闭 LSE AGC 自动增益电路。 1：开启并使用 LSE AGC 自动增益电路。
位 14:10	保留
位 9:8	**SEL_AGC**： 设置 AGC 电路的增益能力。 00：最低档位增益。 … … 11 ：最高档位增益。
位 7	**AUTO_IOP**： 0：IOP[1:0]的 LSE 驱动档位设置值立即生效。 1：LSE 驱动档位将由 2'b11 逐渐降为 IOP[1:0]的设置值。
位 6	**NFBYP**： 0：LSE 时钟信号经过片内滤噪器后被使用。 1：LSE 时钟信号不经过片内滤噪器。
位 5	保留
位 4:3	**IOP_MON**： 当 AUTO_IOP 比特设置为 1 时，IOP[1:0]的设置值不会立即生效；LSE 的驱动档位会由 11 逐渐降为 IOP[1:0]的设置值。在此期间，可通过读取 IOP_MON[1:0]获得当前 LSE 的驱动档位值。
位 2	保留
位 1:0	**IOP**： 设置 LSE 晶振电路驱动能力。 00：最低档位驱动能力。 … … 11：最高档位驱动能力。

6.4.6 BKP 寄存器映像

BKP 寄存器是 16 位的可寻址寄存器。

表 11 BKP 寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	保留																																
004h	BKP_DR1	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
008h	BKP_DR2	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	BKP_DR3	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	BKP_DR4	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	BKP_DR5	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	BKP_DR6	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01Ch	BKP_DR7	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	BKP_DR8	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
024h	BKP_DR9	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028h	BKP_DR10	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02Ch	BKP_RTCCR	保留																						ASOS	ASOE	CCO	CAL[6:0]						
	复位值																							0	0	0	0	0	0	0	0	0	0
030h	RTC_CR	保留																													TPAL	TPE
	复位值																														0	0
034h	RTC_CSR	保留																						TIF	TFE	保留				TPIE	CTI	CTE
	复位值																							0	0					0	0	0
038h	保留																																
03Ch	保留																																
040h	BKP_DR11	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
044h	BKP_DR12	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
048h	BKP_DR13	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04Ch	BKP_DR14	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
050h	BKP_DR15	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
054h	BKP_DR16	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
058h	BKP_DR17	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05Ch	BKP_DR18	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
060h	BKP_DR19	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
064h	BKP_DR20	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
068h	BKP_DR21	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
06Ch	BKP_DR22	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
070h	BKP_DR23	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
074h	BKP_DR24	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
078h	BKP_DR25	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07Ch	BKP_DR26	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
080h	BKP_DR27	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
084h	BKP_DR28	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
088h	BKP_DR29	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08Ch	BKP_DR30	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
090h	BKP_DR31	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
094h	BKP_DR32	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
098h	BKP_DR33	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
09Ch	BKP_DR34	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0A0h	BKP_DR35	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0A4h	BKP_DR36	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0A8h	BKP_DR37	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0ACh	BKP_DR38	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0B0h	BKP_DR39	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0B4h	BKP_DR40	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0B8h	BKP_DR41	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0BCh	BKP_DR42	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器的起始地址，请参考表 1。

7 复位和时钟控制（RCC）

7.1 复位

HK32F10xxx 支持三种复位形式，分别为系统复位、上电复位和备份区域复位。

7.1.1 系统复位

除了时钟控制器的 RCC_CSR 寄存器中的复位标志位和备份区域中的寄存器(见图 3)以外，系统复位将复位所有寄存器至它们的复位状态。

当发生以下任一事件时，产生一个系统复位：

1. NRST 引脚上的低电平(外部复位)
2. 窗口看门狗计数终止(WWDG 复位)
3. 独立看门狗计数终止(IWDG 复位)
4. 软件复位(SW 复位)
5. 低功耗管理复位

可通过查看 RCC_CSR 控制状态寄存器中的复位状态标志位识别复位事件来源。

软件复位

通过将 Cortex™-M3 中断应用和复位控制寄存器中的 SYSRESETREQ 位置'1'，可实现软件复位。请参考 Cortex™-M3 技术参考手册获得进一步信息。

低功耗管理复位

在以下两种情况下可产生低功耗管理复位：

1. 在进入待机模式时产生低功耗管理复位：通过将用户选择字节中的 nRST_STDBY 位置'1'将使能该复位。这时，即使执行了进入待机模式的过程，系统将被复位而不是进入待机模式。
2. 在进入停止模式时产生低功耗管理复位：通过将用户选择字节中的 nRST_STOP 位置'1'将使能该复位。这时，即使执行了进入停机模式的过程，系统将被复位而不是进入停机模式。

7.1.2 电源复位

当以下事件中之一发生时，产生电源复位：

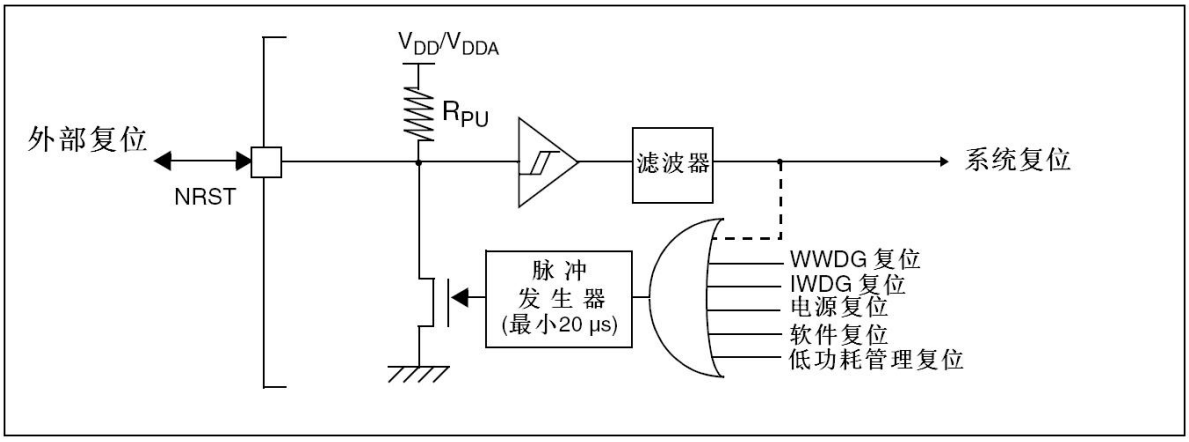
1. 上电/掉电复位(POR/PDR 复位)
2. 从待机模式中返回

电源复位将复位除了备份区域外的所有寄存器。(见图 3)

图中复位源将最终作用于 RESET 引脚，并在复位过程中保持低电平。复位入口矢量被固定在地址 0x0000_0004。更多细节，参阅表 37：其它 HK32F10xxx 产品的向量表。

芯片内部的复位信号会在 NRST 引脚上输出，脉冲发生器保证每一个(外部或内部)复位源都能有至少 40μs 的脉冲延时；当 NRST 引脚被拉低产生外部复位时，它将产生复位脉冲。

图 6 复位电路



7.1.3 备份域复位

备份区域拥有两个专门的复位，它们只影响备份区域(见图 3)。

当以下事件中之一发生时，产生备份区域复位。

1. 软件复位，由设置备份域控制寄存器 (RCC_BDCR)(见 8.3.9 节)中的 BDRST 位产生。
2. 在 V_{DD} 和 V_{BAT} 两者掉电的前提下，V_{DD} 或 V_{BAT} 上电将引发备份区域复位。

7.2 时钟

三种不同的时钟源可被用来驱动系统时钟(SYSCLK)：

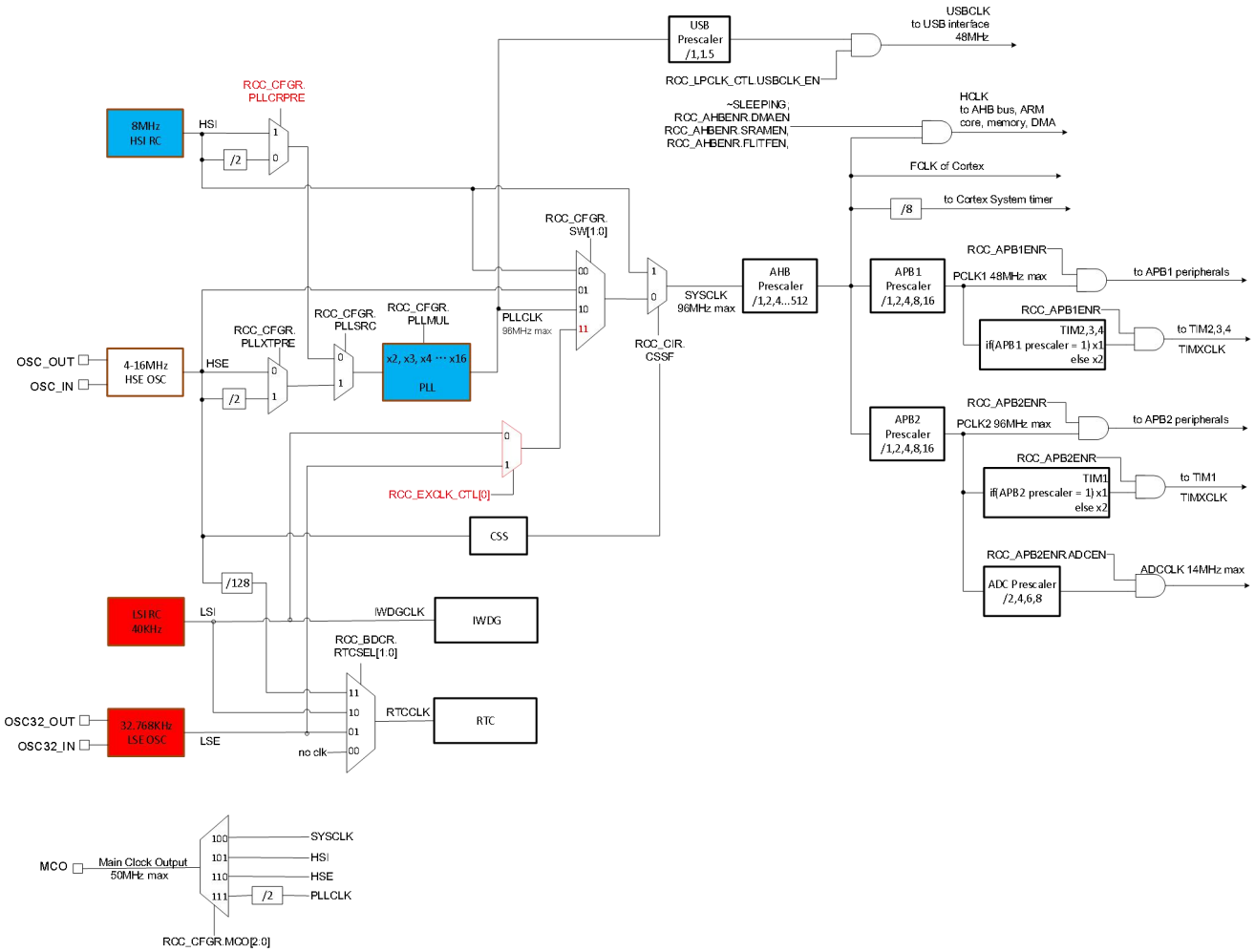
- HSI 振荡器时钟
- HSE 振荡器时钟
- PLL 时钟

这些设备有以下 2 种二级时钟源：

- 40kHz 低速内部 RC，可以用于驱动独立看门狗和通过程序选择驱动 RTC。RTC 用于从停机/待机模式下自动唤醒系统。
- 32.768kHz 低速外部晶体也可用来通过程序选择驱动 RTC(RTCCLK)。

当不被使用时，任一个时钟源都可被独立地启动或关闭，由此优化系统功耗。

图 7 时钟树



1. HK32F103 **支持将** 32KHz LSE **或者** 40KHz LSI **作为系统时钟；如上图红色部分所示。**
2. HK32F103 **支持** HSI **时钟直接作为** PLL **的输入参考时钟，使得最高工作频率达到 96MHz；如上图蓝色部分所示。**

4. **对于内部和外部时钟源的特性，请参考相应产品数据手册中“电气特性”章节。**

用户可通过多个预分频器配置 AHB、高速 APB(APB2)和低速 APB(APB1)域的频率。AHB 和APB2 域的最大频率是 96MHz。APB1 域的最大允许频率是 48MHz。

RCC 通过 AHB 时钟(HCLK) 8 分频后作为 Cortex 系统定时器(SysTick)的外部时钟。通过对 SysTick控制与状态寄存器的设置，可选择上述时钟或 Cortex(HCLK)时钟作为 SysTick 时钟。

ADC 时钟由高速 APB2 时钟经 2、4、6 或 8 分频后获得。

定时器时钟频率分配由硬件按以下 2 种情况自动设置：

1. 如果相应的 APB 预分频系数是 1，定时器的时钟频率与所在 APB 总线频率一致。
2. 否则，定时器的时钟频率被设为与其相连的 APB 总线频率的 2 倍。

FCLK 是 Cortex™-M3 的自由运行时钟。详情见 ARM 的 Cortex™-M3 技术参考手册。

7.2.1 HSE 时钟

高速外部时钟信号(HSE)由以下两种时钟源产生：

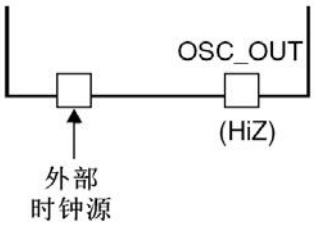
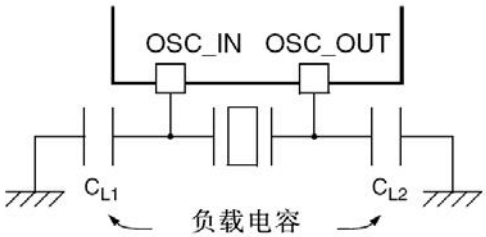
- HSE 外部晶体/陶瓷谐振器
- HSE 用户外部时钟

为了减少时钟输出的失真和缩短启动稳定时间，晶体/陶瓷谐振器和负载电容器必须尽可能地靠近振荡器引脚。负载电容值必须根据所选择的振荡器来调整。

HK32F103 的 HSE 具有以下特性：

1. 可以选择通过片内滤噪器后再被芯片内部电路使用，以增强HSE时钟对环境噪声的抵抗能力。
2. HSE晶振振荡稳定等待时间可配。
3. HSE晶振引脚驱动能力可配。

图 8 HSE/LSE 时钟源

	硬件配置
外部时钟	
晶体/陶瓷谐振器	

外部时钟源(HSE 旁路)

在这个模式里，必须提供外部时钟。它的频率最高可达 25MHz。用户可通过设置在时钟控制寄存器中的 HSEBYP 和 HSEON 位来选择这一模式。外部时钟信号(50%占空比的方波、正弦波或三角波)必须连到 SOC_IN 引脚，同时保证 OSC_OUT 引脚悬空。见图 8。

外部晶体/陶瓷谐振器(HSE 晶体)

4~16Mz 外部振荡器可为系统提供更为精确的主时钟。相关的硬件配置可参考图 8，进一步信息可参考数据手册的电气特性部分。

在时钟控制寄存器 RCC_CR 中的 HSERDY 位用来指示高速外部振荡器是否稳定。在启动时，直到这一位被硬件置'1'，时钟才被释放出来。如果在时钟中断寄存器 RCC_CIR 中允许产生中断，将会产生相应中断。

HSE 晶体可以通过设置时钟控制寄存器里 RCC_CR 中的 HSEON 位被启动和关闭。

7.2.2 HSI 时钟

HSI 时钟信号由内部 8MHz 的 RC 振荡器产生，可直接作为系统时钟或作为 PLL 输入。

HSI RC 振荡器能够在不需要任何外部器件的条件下提供系统时钟。它的启动时间比 HSE 晶体振荡

器短。然而，即使在校准之后它的时钟频率精度仍较差。**校准**

制造工艺决定了不同芯片的 RC 振荡器频率会不同，这就是为什么每个芯片的 HSI 时钟频率在出厂前已经被校准到 1%(25°C)的原因。系统复位时，工厂校准值被装载到时钟控制寄存器的HSICAL[7:0]位。

如果用户的应用基于不同的电压或环境温度，这将会影响 RC 振荡器的精度。可以通过时钟控制寄存器里的 HSITRIM[4:0]位来调整 HSI 频率。时钟控制寄存器中的 HSIRDY 位用来指示 HSI RC 振荡器是否稳定。在时钟启动过程中，直到这一位被硬件置'1'，HSI RC 输出时钟才被释放。HSI RC 可由时钟控制寄存器中的 HSION 位来启动和关闭。

如果 HSE 晶体振荡器失效，HSI 时钟会被作为备用时钟源。参考 8.2.7 节时钟安全系统。

7.2.3 PLL

内部 PLL 可以用来倍频 HSI RC 的输出时钟或 HSE 晶体输出时钟。参考图 8 和时钟控制寄存器。

PLL 的设置(选择 HSI 或者 HSI/2 或 HSE 时钟为 PLL 的输入时钟，和选择倍频因子)必须在其被激活前完成。一旦 PLL 被激活，这些参数就不能被改动。

如果 PLL 中断在时钟中断寄存器里被允许，当 PLL 准备就绪时，可产生中断申请。如果需要在应用中使用 USB 接口，PLL 必须被设置为输出 48 或 72MHZ 时钟，用于提供 48MHz 的 USBCLK 时钟。

7.2.4 LSE 时钟

LSE 晶体是一个 32.768kHz 的低速外部晶体或陶瓷谐振器。它为实时时钟或者其他定时功能提供一个低功耗且精确的时钟源。

HK32F103 的 LSE 具有以下的特性：

1. 可以选择通过片内滤噪器后再被芯片内部电路使用，以增强LSE时钟对环境噪声的抵抗能力。
2. LSE晶振引脚驱动能力可配。
3. HK32F103的LSE晶振电路带有AGC自动增益；通过使能AGC电路可显著降低LSE晶振电流功耗。

LSE 晶体通过在备份域控制寄存器(RCC_BDCR)里的 LSEON 位启动和关闭。

在备份域控制寄存器(RCC_BDCR)里的 LSERDY 指示 LSE 晶体振荡是否稳定。在启动阶段，直到这个位被硬件置'1'后，LSE 时钟信号才被释放出来。如果在时钟中断寄存器里被允许，可产生中断申请。

外部时钟源(LSE 旁路)

在这个模式里必须提供一个 32.768kHz 频率的外部时钟源。你可以通过设置在备份域控制寄存器(RCC_BDCR)里的 LSEBYP 和 LSEON 位来选择这个模式。具有 50%占空比的外部时钟信号(方波、正弦波或三角波)必须连到 OSC32_IN 引脚，同时保证 OSC32_OUT 引脚悬空，见图 8。

7.2.5 LSI 时钟

LSI RC 担当一个低功耗时钟源的角色，它可以在停机和待机模式下保持运行，为独立看门狗和自动唤醒单元提供时钟。LSI 时钟频率大约 40kHz(在 30kHz 和 60kHz 之间)。进一步信息请参考数据手册中有关电气特性部分。

LSI RC 可以通过控制/状态寄存器(RCC_CSR)里的 LSION 位来启动或关闭。

在控制/状态寄存器(RCC_CSR)里的 LSIRDY 位指示低速内部振荡器是否稳定。在启动阶段，直到这个位被硬件设置为'1'后，此时钟才被释放。如果在时钟中断寄存器(RCC_CIR)里被允许，将产生 LSI 中断申请。

7.2.6 系统时钟(SYSCLK)选择

系统复位后，HSI 振荡器被选为系统时钟。当时钟源被直接或通过 PLL 间接作为系统时钟时，它将不能被停止。

只有当目标时钟源准备就绪了(经过启动稳定阶段的延迟或 PLL 稳定)，从一个时钟源到另一个时钟源的切换才会发生。在被选择时钟源没有就绪时，系统时钟的切换不会发生。直至目标时钟源就绪，才发生切换。

在时钟控制寄存器(RCC_CR)里的状态位指示哪个时钟已经准备好了，哪个时钟目前被用作系统时钟。

7.2.7 时钟安全系统(CSS)

时钟安全系统可以通过软件被激活。一旦其被激活，时钟监测器将在 HSE 振荡器启动延迟后被使能，并在 HSE 时钟关闭后关闭。

如果 HSE 时钟发生故障，HSE 振荡器被自动关闭，时钟失效事件将被送到高级定时器(TIM1)的刹车输入端，并产生时钟安全中断 CSSI，允许软件完成营救操作。此 CSSI 中断连接到 Cortex™-M3 的 NMI 中断(不可屏蔽中断)。

注意：一旦 CSS 被激活，并且 HSE 时钟出现故障，CSS 中断就产生，并且 NMI 也自动产生。NMI 将被不断执行，直到 CSS 中断挂起位被清除。因此，在 NMI 的处理程序中必须通过设置时钟中断寄存器(RCC_CIR)里的 CSSC 位来清除 CSS 中断。

如果 HSE 振荡器被直接或间接地作为系统时钟，(间接的意思是：它被作为 PLL 输入时钟，并且PLL 时钟被作为系统时钟)，时钟故障将导致系统时钟自动切换到 HSI 振荡器，同时外部 HSE 振荡 器被关闭。在时钟失效时，如果 HSE 振荡器时钟(被分频或未被分频)是用作系统时钟的 PLL 的输入时钟，PLL 也将被关闭。

7.2.8 RTC 时钟

通过设置备份域控制寄存器(RCC_BDCR)里的 RTCSEL[1:0]位，RTCCLK 时钟源可以由 HSE/128、LSE 或 LSI 时钟提供。除非备份域复位，此选择不能被改变。 LSE 时钟在备份域里，但 HSE 和 LSI 时钟不是。因此：

- 如果 LSE 被选为 RTC 时钟：
  - 只要 V_{BAT} 维持供电，尽管 V_{DD} 供电被切断，RTC 仍继续工作。
- 如果 LSI 被选为自动唤醒单元(AWU)时钟：
  - 如果 V_{DD} 供电被切断，AWU 状态不能被保证。
- 如果 HSE 时钟 128 分频后作为 RTC 时钟：
  - 如果 V_{DD} 供电被切断或内部电压调压器被关闭(1.8V 域的供电被切断)，则 RTC 状态不确定。
  - 必须设置电源控制寄存器(见 6.4.1 节)的 DPB 位(取消后备区域的写保护)为’1’。

7.2.9 看门狗时钟

如果独立看门狗已经由硬件选项或软件启动，LSI 振荡器将被强制在打开状态，并且不能被关闭。在 LSI 振荡器稳定后，时钟供应给 IWDG。

7.2.10 时钟输出

微控制器允许输出时钟信号到外部 MCO 引脚。相应的 GPIO 端口寄存器必须被配置为相应功能。以下四个时钟信号可被选作 MCO 时钟：

- SYSCLK
- HSI
- HSE

- 除 2 的 PLL 时钟

时钟的选择由时钟配置寄存器(RCC_CFGR)中的 MCO[2:0]位控制。

7.3 RCC 寄存器描述

请参考第 2 章中有关寄存器描述中用到的缩写。

7.3.1 时钟控制寄存器(RCC_CR)

偏移地址: 0x00

复位值: 0x000 XX83，X 代表未定义

访问: 无等待状态,字,半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留						PLL RDY	PLLON	保留				CSS ON	HSE BYP	HSE RDY	HSE ON
						r	rw					rw	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					保留	HSI RDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

位 31:26	保留，始终读为 0。
位 25	**PLLRDY**：PLL 时钟就绪标志(PLL clock ready flag) PLL 锁定后由硬件置'1'。 0：PLL 未锁定； 1：PLL 锁定。
位 24	**PLLON**：PLL 使能(PLL enable) 由软件置'1'或清零。 当进入待机和停止模式时，该位由硬件清零。当 PLL 时钟被用作或被选择将要作为系统时时，该位不能被清零。 0：PLL 关闭； 1：PLL 使能。
位 23:20	保留，始终读为 0。
位 19	**CSSON**：时钟安全系统使能(Clock security system enable) 由软件置'1'或清零以使能时钟监测器。 0：时钟监测器关闭； 1：如果外部 4-16MHz 振荡器就绪，时钟监测器开启。
位 18	**HSEBYP**：外部高速时钟旁路(External high-speed clock bypass) 在调试模式下由软件置'1'或清零来旁路外部晶体振荡器。只有在外部 4-16MHz 振荡器关闭的情况下，才能写入该位。 0：外部 4-16MHz 振荡器没有旁路； 1：外部 4-16MHz 外部晶体振荡器被旁路。
位 17	**HSERDY**：外部高速时钟就绪标志(External high-speed clock ready flag) 由硬件置'1'来指示外部 4-16MHz 振荡器已经稳定。在 HSEON 位清零后，该位需要 6 个外部 25MHz 振荡器周期清零。 0：外部 4-16MHz 振荡器没有就绪； 1：外部 4-16MHz 振荡器就绪。

位 16	**HSEON**：外部高速时钟使能(External high-speed clock enable) 由软件置'1'或清零。 当进入待机和停止模式时，该位由硬件清零，关闭 4-16MHz 外部振荡器。当外部 4-16MHz 振荡器被用作或被选择将要作为系统时钟时，该位不能被清零。 0：HSE 振荡器关闭； 1：HSE 振荡器开启。
位 15:8	**HSICAL[7:0]**：内部高速时钟校准(Internal high-speed clock calibration) 在系统启动时，这些位被自动初始化
位 7:3	**HSITRIM[4:0]**：内部高速时钟调整(Internal high-speed clock trimming) 由软件写入来调整内部高速时钟，它们被叠加在 HSICAL[5:0]数值上。 这些位在 HSICAL[7:0]的基础上，让用户可以输入一个调整数值，根据电压和温度的变化调整内部 HSI RC 振荡器的频率。 默认数值为 16，可以把 HSI 调整到 8MHz±1%；每步 HSICAL 的变化调整约 40kHz。
位 2	保留，始终读为 0。
位 1	**HSIRDY**：内部高速时钟就绪标志(Internal high-speed clock ready flag) 由硬件置'1'来指示内部 8MHz 振荡器已经稳定。在 HSION 位清零后，该位需要 6 个内部 8MHz 振荡器周期清零。 0：内部 8MHz 振荡器没有就绪； 1：内部 8MHz 振荡器就绪。
位 0	**HSION**：内部高速时钟使能(Internal high-speed clock enable) 由软件置'1'或清零。 当从待机和停止模式返回或用作系统时钟的外部 4-16MHz 振荡器发生故障时，该位由硬件置'1'来启动内部 8MHz 的 RC 振荡器。当内部 8MHz 振荡器被直接或间接地用作或被选择将要作为系统时钟时，该位不能被清零。 0：内部 8MHz 振荡器关闭； 1：内部 8MHz 振荡器开启。

7.3.2 时钟配置寄存器(RCC_CFGR)

偏移地址: 0x04

复位值: 0x0000 0000

访问: 0 到 2 个等待周期，字，半字和字节访问只有当访问发生在时钟切换时，才会插入 1 或 2 个等待周期。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLLHSI PRE	保留				MCO[2:0]			保留	USB PRE	PLLMUL[3:0]				PLL XTPR	PLL SRC
					rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCPRE[1:0]		PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

位 31	**RCC_CFGR[31]** ： PLLHSIPRE 1'b0 : 选择 HSI 的 2 分频时钟作为 PLL 的输入参考时钟 1'b1 : 选择 HSI 时钟作为 PLL 的输入参考时钟
位 30：27	保留，始终读为 0。

位 26:24	**MCO**：微控制器时钟输出(Microcontroller clock output) 由软件置'1'或清零。 0xx：没有时钟输出； 100：系统时钟(SYSCLK)输出； 101：内部 RC 振荡器时钟(HSI)输出； 110：外部振荡器时钟(HSE)输出； 111：PLL 时钟 2 分频后输出。 注意：- 该时钟输出在启动和切换 MCO 时钟源时可能会被截断。 - 在系统时钟作为输出至 MCO 引脚时，请保证输出时钟频率不超过 50MHz (I/O 口最高频率)。
位 22	**USBPRE**：USB 预分频(USB prescaler) 由软件置'1'或清'0'来产生 48MHz 的 USB 时钟。在 RCC_APB1ENR 寄存器中使能 USB 时钟之前，必须保证该位已经有效。如果 USB 时钟被使能，该位不能被清零。 0：PLL 时钟 1.5 倍分频作为 USB 时钟 1：PLL 时钟直接作为 USB 时钟
位 21:18	**PLLMUL**：PLL 倍频系数(PLL multiplication factor) 由软件设置来确定 PLL 倍频系数。只有在 PLL 关闭的情况下才可被写入。 注意：PLL 的输出频率不能超过 96MHz 0000：PLL 2 倍频输出　1000：PLL 10 倍频输出 0001：PLL 3 倍频输出　1001：PLL 11 倍频输出 0010：PLL 4 倍频输出　1010：PLL 12 倍频输出 0011：PLL 5 倍频输出　1011：PLL 13 倍频输出 0100：PLL 6 倍频输出　1100：PLL 14 倍频输出 0101：PLL 7 倍频输出　1101：PLL 15 倍频输出 0110：PLL 8 倍频输出　1110：PLL 16 倍频输出 0111：PLL 9 倍频输出　1111：PLL 16 倍频输出
位 17	**PLLXTPRE**：HSE 分频器作为 PLL 输入(HSE divider for PLL entry) 由软件置'1'或清'0'来分频 HSE 后作为 PLL 输入时钟。只能在关闭 PLL 时才能写入此位。 0：HSE 不分频 1：HSE 2 分频
位 16	**PLLSRC**：PLL 输入时钟源(PLL entry clock source) 由软件置'1'或清'0'来选择 PLL 输入时钟源。只能在关闭 PLL 时才能写入此位。 0：HSI 时钟作为 PLL 输入时钟 1：HSE 时钟作为 PLL 输入时钟。
位 15:14	**ADCPRE[1:0]**：ADC 预分频(ADC prescaler) 由软件置'1'或清'0'来确定 ADC 时钟频率 00： PCLK2 2 分频后作为 ADC 时钟 01：PCLK2 4 分频后作为 ADC 时钟 10：PCLK2 6 分频后作为 ADC 时钟 11：PCLK2 8 分频后作为 ADC 时钟
位 13:11	**PPRE2[2:0]**：高速 APB 预分频(APB2) (APB high-speed prescaler (APB2)) 由软件置'1'或清'0'来控制高速 APB2 时钟(PCLK2)的预分频系数。 0xx：HCLK 不分频 100：HCLK 2 分频 101：HCLK 4 分频 110：HCLK 8 分频 111：HCLK 16 分频

位 10:8	**PPRE1[2:0]**：低速 APB 预分频(APB1) (APB low-speed prescaler (APB1)) 由软件置'1'或清'0'来控制低速 APB1 时钟(PCLK1)的预分频系数。 警告：软件必须保证 APB1 时钟频率不超过 36MHz。 0xx：HCLK 不分频 100：HCLK 2 分频 101：HCLK 4 分频 110：HCLK 8 分频 111：HCLK 16 分频
位 7:4	**HPRE[3:0]**：AHB 预分频(AHB Prescaler) 由软件置'1'或清'0'来控制 AHB 时钟的预分频系数。 0xxx：SYSCLK 不分频 1000：SYSCLK 2 分频　　1100：SYSCLK 64 分频 1001：SYSCLK 4 分频　　1101：SYSCLK 128 分频 1010：SYSCLK 8 分频　　1110：SYSCLK 256 分频 1011：SYSCLK 16 分频　　1111：SYSCLK 512 分频 注意：当 AHB 时钟的预分频系数大于 1 时，必须开启预取缓冲器。详见闪存读取(第 4.3.3 节)。
位 3:2	**SWS[1:0]**：系统时钟切换状态(System clock switch status) 由硬件置'1'或清'0'来指示哪一个时钟源被作为系统时钟。 00：HSI 作为系统时钟； 01：HSE 作为系统时钟； 10：PLL 输出作为系统时钟； 11：LSE 或者 LSI 被选择成为系统时钟。
位 1:0	**SW[1:0]**：系统时钟切换(System clock switch) 由软件置'1'或清'0'来选择系统时钟源。 在从停止或待机模式中返回时或直接或间接作为系统时钟的 HSE 出现故障时，由硬件强制选择 HSI 作为系统时钟(如果时钟安全系统已经启动) 00：HSI 作为系统时钟； 01：HSE 作为系统时钟； 10：PLL 输出作为系统时钟； 11：选择 LSE 或者 LSI 作为系统时钟。

7.3.3 时钟中断寄存器(RCC_CIR)

偏移地址: 0x08

复位值: 0x0000 0000

访问:无等待周期，字，半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留								CSSC	保留		PLL RDYC	HSE RDYC	HIS RDYC	LSE RDY	LSI RDYC
								w			w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			PLL RDYI	HSE RDYI	HSI RDYI	LSE RDYI	LSI RDYI	CSSF	保留		PLL RDY	HSE RDY	HSI RDY	LSE RDY	LSI RDY
			rw	rw	rw	rw	rw	r			r	r	r	r	r

位 31:24	保留，始终读为 0。

位 23	**CSSC**：清除时钟安全系统中断(Clock security system interrupt clear) 由软件置'1'来清除 CSSF 安全系统中断标志位 CSSF。 0：无作用； 1：清除 CSSF 安全系统中断标志位。

位 22:21	保留，始终读为 0。
位 20	**PLLRDYC**：清除 PLL 就绪中断(PLL ready interrupt clear) 由软件置'1'来清除 PLL 就绪中断标志位 PLLRDYF。 0：无作用； 1：清除 PLL 就绪中断标志位 PLLRDYF。
位 19	**HSERDYC**：清除 HSE 就绪中断(HSE ready interrupt clear) 由软件置'1'来清除 HSE 就绪中断标志位 HSERDYF。 0：无作用； 1：清除 HSE 就绪中断标志位 HSERDYF。
位 18	**HSIRDYC**：清除 HSI 就绪中断(HSI ready interrupt clear) 由软件置'1'来清除 HSI 就绪中断标志位 HSIRDYF。 0：无作用； 1：清除 HSI 就绪中断标志位 HSIRDYF。
位 17	**LSERDYC**：清除 LSE 就绪中断(LSE ready interrupt clear) 由软件置'1'来清除 LSE 就绪中断标志位 LSERDYF。 0：无作用； 1：清除 LSE 就绪中断标志位 LSERDYF。
位 16	**LSIRDYC**：清除 LSI 就绪中断(LSI ready interrupt clear) 由软件置'1'来清除 LSI 就绪中断标志位 LSIRDYF。 0：无作用； 1：清除 LSI 就绪中断标志位 LSIRDYF。
位 15:13	保留，始终读为 0。
位 12	**PLLRDYIE**：PLL 就绪中断使能(PLL ready interrupt enable) 由软件置'1'或清'0'来使能或关闭 PLL 就绪中断。 0：PLL 就绪中断关闭； 1：PLL 就绪中断使能。
位 11	**HSERDYIE**：HSE 就绪中断使能(HSE ready interrupt enable) 由软件置'1'或清'0'来使能或关闭外部 4-16MHz 振荡器就绪中断。 0：HSE 就绪中断关闭； 1：HSE 就绪中断使能。
位 10	**HSIRDYIE**：HSI 就绪中断使能(HSI ready interrupt enable) 由软件置'1'或清'0'来使能或关闭内部 8MHz RC 振荡器就绪中断。 0：HSI 就绪中断关闭； 1：HSI 就绪中断使能。
位 9	**LSERDYIE**：LSE 就绪中断使能(LSE ready interrupt enable) 由软件置'1'或清'0'来使能或关闭外部 32kHz RC 振荡器就绪中断。 0：LSE 就绪中断关闭； 1：LSE 就绪中断使能。
位 8	**LSIRDYIE**：LSI 就绪中断使能(LSI ready interrupt enable) 由软件置'1'或清'0'来使能或关闭内部 40kHz RC 振荡器就绪中断。 0：LSI 就绪中断关闭； 1：LSI 就绪中断使能。

位 7	**CSSF**：时钟安全系统中断标志(Clock security system interrupt flag) 在外部 4-16MHz 振荡器时钟出现故障时，由硬件置'1'。 由软件通过置'1' CSSC 位来清除。 0：无 HSE 时钟失效产生的安全系统中断； 1：HSE 时钟失效导致了时钟安全系统中断。
位 6:5	保留，始终读为 0。

位 4	**PLLRDYF**：PLL 就绪中断标志(PLL ready interrupt flag) 在 PLL 就绪且 PLLRDYIE 位被置'1'时，由硬件置'1'。 由软件通过置'1' PLLRDYC 位来清除。 0：无 PLL 上锁产生的时钟就绪中断； 1：PLL 上锁导致时钟就绪中断。
位 3	**HSERDYF**：HSE 就绪中断标志(HSE ready interrupt flag) 在外部低速时钟就绪且 HSERDYIE 位被置'1'时，由硬件置'1'。 由软件通过置'1' HSERDYC 位来清除。 0：无外部 4-16MHz 振荡器产生的时钟就绪中断； 1：外部 4-16MHz 振荡器导致时钟就绪中断。
位 2	**HSIRDYF**：HSI 就绪中断标志(HSI ready interrupt flag) 在内部高速时钟就绪且 HSIRDYIE 位被置'1'时，由硬件置'1'。 由软件通过置'1' HSIRDYC 位来清除。 0：无内部 8MHz RC 振荡器产生的时钟就绪中断； 1：内部 8MHz RC 振荡器导致时钟就绪中断。
位 1	**LSERDYF**：LSE 就绪中断标志(LSE ready interrupt flag) 在外部低速时钟就绪且 LSERDYIE 位被置'1'时，由硬件置'1'。 由软件通过置'1' LSERDYC 位来清除。 0：无外部 32kHz 振荡器产生的时钟就绪中断； 1：外部 32kHz 振荡器导致时钟就绪中断。
位 0	**LSIRDYF**：LSI 就绪中断标志(LSI ready interrupt flag) 在内部低速时钟就绪且 LSIRDYIE 位被置'1'时，由硬件置'1'。 由软件通过置'1' LSIRDYC 位来清除。 0：无内部 40kHz RC 振荡器产生的时钟就绪中断； 1：内部 40kHz RC 振荡器导致时钟就绪中断。

7.3.4 APB2 外设复位寄存器 (RCC_APB2RSTR)

偏移地址: 0x0C

复位值: 0x0000 0000

访问：无等待周期，字，半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	USART1RST	-	SPI1RST	TIM1RST	ADC2RST	ADC1RST	IOPG	IOPFRST	IOPERST	IOPDRST	IOPCRST	IOPBRST	IOPARST	保留	AFIORST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res	rw

位 31:15	保留，始终读为 0。

位 14	**USART1RST**：USART1 复位(USART1 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 USART1。
位 13	保留
位 12	**SPI1RST**：SPI1 复位(SPI 1 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 SPI1。

位 11	**TIM1RST**：TIM1 定时器复位(TIM1 timer reset) 由软件置'1'或清'0' 0：无作用； 1：复位 TIM1 定时器。
位 10	**ADC2RST**：ADC2 接口复位(ADC 2 interface reset) 由软件置'1'或清'0' 0：无作用； 1：复位 ADC2 接口。
位 9	**ADC1RST**：ADC1 接口复位(ADC 1 interface reset) 由软件置'1'或清'0' 0：无作用； 1：复位 ADC1 接口。
位 8	**IOPGRST**：IO 端口 G 复位(IO port G reset) 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 G。
位 7	**IOPFRST**：IO 端口 F 复位(IO port F reset) 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 F。
位 6	**IOPERST**：IO 端口 E 复位(IO port E reset) 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 E。
位 5	**IOPDRST**：IO 端口 D 复位(IO port D reset) 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 D。
位 4	**IOPCRST**：IO 端口 C 复位(IO port C reset) 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 C。
位 3	**IOPBRST**：IO 端口 B 复位(IO port B reset) 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 B。

位 2	**IOPARST**：IO 端口 A 复位(IO port A reset) 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 A。
位 1	保留，始终读为 0。
位 0	**AFIORST**：辅助功能 IO 复位(Alternate function I/O reset) 由软件置'1'或清'0' 0：无作用； 1：复位辅助功能。

7.3.5 APB1 外设复位寄存器 (RCC_APB1RSTR)

偏移地址：0x10

复位值：0x0000 0000

访问：无等待周期，字，半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留		DACRST	PWRRST	BKPRST	保留	CANRST	保留	USBRST	I2C2RST	I2C1RST	-	-	USART3RST	USART2RST	保留
		rw	rw	rw		rw		rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3RST	SPI2RST	保留		WWDGRST	保留								TIM4RST	TIM3RST	TIM2RST
rw	rw			rw									rw	rw	rw

位 31:30	保留，始终读为 0。
位 29	**DACRST**：DAC 接口复位(DAC interface reset) 由软件置'1'或清'0' 0：无作用； 1：复位 DAC 接口。
位 28	**PWRRST**：电源接口复位(Power interface reset) 由软件置'1'或清'0' 0：无作用； 1：复位电源接口。
位 27	**BKPRST**：备份接口复位(Backup interface reset) 由软件置'1'或清'0' 0：无作用； 1：复位备份接口。
位 26	保留，始终读为 0。
位 25	**CANRST**：CAN 复位(CAN reset) 由软件置'1'或清'0' 0：无作用； 1：复位 CAN。
位 24	保留，始终读为 0。
位 23	**USBRST**：USB 复位(USB reset) 由软件置'1'或清'0' 0：无作用； 1：复位 USB。

位 22	**I2C2RST**：I2C 2 复位(I2C 2 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 I2C 2
位 21	**I2C1RST**：I2C 1 复位(I2C 1 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 I2C 1。
位 20	–
位 19	–
位 18	**USART3RST**：USART3 复位(USART 3 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 USART3。
位 17	**USART2RST**：USART2 复位(USART 2 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 USART2。
位 16	保留，始终读为 0。
位 15	**SPI3RST** SPI3 复位(SPI 3 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 SPI3。
位 14	**SPI2RST**：SPI2 复位(SPI 2 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 SPI2。
位 13:12	保留，始终读为 0。
位 11	**WWDGRST**：窗口看门狗复位(Window watchdog reset) 由软件置'1'或清'0' 0：无作用； 1：复位窗口看门狗。
位 10:6	保留，始终读为 0。
位 5	保留。
位 4	保留
位 3	保留
位 2	**TIM4RST**：定时器 4 复位(Timer 4 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 TIM4 定时器。

位 1	**TIM3RST**：定时器 3 复位(Timer 3 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 TIM3 定时器。
位 0	**TIM2RST**：定时器 2 复位(Timer 2 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 TIM2 定时器。

7.3.6 AHB 外设时钟使能寄存器 (RCC_AHBENR)

偏移地址：0x14
复位值：0x0000 0014
访问：无等待周期,字,半字和字节访问

注：　当外设时钟没有启用时，软件不能读出外设寄存器的数值，返回的数值始终是 0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					保留	保留	保留	保留	CRCEN	保留	FLITFEN	保留	SRAMEN	保留	DMA1EN
									rw		rw		rw		rw

位 31:11	保留，始终读为 0。
位 10	保留
位 9	保留，始终读为 0。
位 8	保留
位 7	保留，始终读为 0。
位 6	**CRCEN**：CRC 时钟使能(CRC clock enable) 由软件置'1'或清'0'。 0：CRC 时钟关闭； 1：CRC 时钟开启。
位 5	保留，始终读为 0。
位 4	**FLITFEN**：闪存接口电路时钟使能(FLITF clock enable) 由软件置'1'或清'0'来开启或关闭睡眠模式时闪存接口电路时钟。 0：睡眠模式时闪存接口电路时钟关闭； 1：睡眠模式时闪存接口电路时钟开启。
位 3	保留，始终读为 0。
位 2	**SRAMEN**：SRAM 时钟使能(SRAM interface clock enable) 由软件置'1'或清'0'来开启或关闭睡眠模式时 SRAM 时钟。 0：睡眠模式时 SRAM 时钟关闭； 1：睡眠模式时 SRAM 时钟开启。

位 1	保留
位 0	**DMA1EN**：DMA1 时钟使能(DMA1 clock enable) 由软件置'1'或清'0'。 0：DMA1 时钟关闭； 1：DMA1 时钟开启。

7.3.7 APB2 外设时钟使能寄存器(RCC_APB2ENR)

偏移地址：0x18

复位值：0x0000 0000

访问：字，半字和字节访问

通常无访问等待周期。但在 APB2 总线上的外设被访问时，将插入等待状态直到 APB2 的外设访问结束。

注：　当外设时钟没有启用时，软件不能读出外设寄存器的数值，返回的数值始终是 0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	USART1 EN	-	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	IOPG EN	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	保留	AFIO EN
	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

位 31:15	保留，始终读为 0。
位 15	保留
位 14	**USART1EN**：USART1 时钟使能(USART1 clock enable) 由软件置'1'或清'0' 0：USART1 时钟关闭； 1：USART1 时钟开启。
位 13	保留
位 12	**SPI1EN**：SPI1 时钟使能(SPI 1 clock enable) 由软件置'1'或清'0' 0：SPI1 时钟关闭； 1：SPI1 时钟开启。
位 11	**TIM1EN**：TIM1 定时器时钟使能(TIM1 Timer clock enable) 由软件置'1'或清'0' 0：TIM1 定时器时钟关闭； 1：TIM1 定时器时钟开启。
位 10	**ADC2EN**：ADC2 接口时钟使能(ADC 2 interface clock enable) 由软件置'1'或清'0' 0：ADC2 接口时钟关闭； 1：ADC2 接口时钟开启。

位 9	**ADC1EN**：ADC1 接口时钟使能(ADC 1 interface clock enable) 由软件置'1'或清'0' 0： ADC1 接口时钟关闭； 1：ADC1 接口时钟开启。
位 8	**IOPGEN**：IO 端口 G 时钟使能(I/O port G clock enable) 由软件置'1'或清'0' 0：IO 端口 G 时钟关闭； 1：IO 端口 G 时钟开启。
位 7	**IOPFEN**：IO 端口 F 时钟使能(I/O port F clock enable) 由软件置'1'或清'0' 0： IO 端口 F 时钟关闭； 1：IO 端口 F 时钟开启。
位 6	**IOPEEN**：IO 端口 E 时钟使能(I/O port E clock enable) 由软件置'1'或清'0' 0： IO 端口 E 时钟关闭； 1：IO 端口 E 时钟开启。
位 5	**IOPDEN**：IO 端口 D 时钟使能(I/O port D clock enable) 由软件置'1'或清'0' 0： IO 端口 D 时钟关闭； 1：IO 端口 D 时钟开启。
位 4	**IOPCEN**：IO 端口 C 时钟使能(I/O port C clock enable) 由软件置'1'或清'0' 0： IO 端口 C 时钟关闭； 1：IO 端口 C 时钟开启。
位 3	**IOPBEN**：IO 端口 B 时钟使能(I/O port B clock enable) 由软件置'1'或清'0' 0： IO 端口 B 时钟关闭； 1：IO 端口 B 时钟开启。
位 2	**IOPAEN**：IO 端口 A 时钟使能(I/O port A clock enable) 由软件置'1'或清'0' 0： IO 端口 A 时钟关闭； 1：IO 端口 A 时钟开启。
位 1	保留，始终读为 0。
位 0	**AFIOEN**：辅助功能 IO 时钟使能(Alternate function I/O clock enable) 由软件置'1'或清'0' 0：辅助 功能 IO 时钟关闭； 1：辅助功能 IO 时钟开启。

7.3.8 APB1 外设时钟使能寄存器(RCC_APB1ENR)

偏移地址：0x1C

复位值：0x0000 0000

访问：字、半字和字节访问

通常无访问等待周期。但在 APB1 总线上的外设被访问时，将插入等待状态直到 APB1 外设访问结束。

注：　当外设时钟没有启用时，软件不能读出外设寄存器的数值，返回的数值始终是 0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留		DACEN	PWR EN	BKP EN	保留	CAN EN	保留	USB EN	I2C2 EN	I2C1 EN	-	-	USART3 EN	USART2 EN	保留
		rw	rw	rw		rw		rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	保留		WWDG EN	保留								TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw									rw	rw	rw

位 31:30	保留，始终读为 0。
位 29	**DACEN**: DAC 接口时钟使能(DAC interface clock enable) 由软件置'1'或清'0' 0：DAC 接口时钟关闭； 1：DAC 接口时钟开启。
位 28	**PWREN**：电源接口时钟使能(Power interface clock enable) 由软件置'1'或清'0' 0：电源接口时钟关闭； 1：电源接口时钟开启。
位 27	**BKPEN**：备份接口时钟使能(Backup interface clock enable) 由软件置'1'或清'0' 0：备份接口时钟关闭； 1：备份接口时钟开启。
位 26	保留，始终读为 0。
位 25	**CANEN**：CAN 时钟使能(CAN clock enable) 由软件置'1'或清'0' 0：CAN 时钟关闭； 1：CAN 时钟开启。
位 24	保留，始终读为 0。
位 23	**USBEN**：USB 时钟使能(USB clock enable) 由软件置'1'或清'0' 0：USB 时钟关闭； 1：USB 时钟开启。
位 22	I2C2EN：I2C 2 时钟使能 (I2C 2 clock enable) 由软件置'1'或清'0' 0：I2C 2 时钟关闭； 1：I2C 2 时钟开启。
位 21	**I2C1EN**：I2C 1 时钟使能(I2C 1 clock enable) 由软件置'1'或清'0' 0：I2C 1 时钟关闭； 1：I2C 1 时钟开启。
位 20	-
位 19	-

位 18	**USART3EN**：USART3 时钟使能(USART 3 clock enable) 由软件置'1'或清'0' 0：USART3 时钟关闭； 1：USART3 时钟开启。
位 17	**USART2EN**：USART2 时钟使能(USART 2 clock enable) 由软件置'1'或清'0' 0：USART2 时钟关闭； 1：USART2 时钟开启。
位 16	保留，始终读为 0。
位 15	**SPI3EN**：SPI 3 时钟使能(SPI 3 clock enable) 由软件置'1'或清'0' 0：SPI 3 时钟关闭； 1：SPI 3 时钟开启。
位 14	**SPI2EN**：SPI 2 时钟使能(SPI 2 clock enable) 由软件置'1'或清'0' 0：SPI 2 时钟关闭； 1：SPI 2 时钟开启。
位 13:12	保留，始终读为 0。
位 11	**WWDGEN**：窗口看门狗时钟使能(Window watchdog clock enable) 由软件置'1'或清'0' 0：窗口看门狗时钟关闭； 1：窗口看门狗时钟开启。
位 10:6	保留，始终读为 0。
位 5	保留
位 4	保留
位 3	保留
位 2	**TIM4EN**：定时器 4 时钟使能(Timer 4 clock enable) 由软件置'1'或清'0' 0：定时器 4 时钟关闭； 1：定时器 4 时钟开启。
位 1	**TIM3EN**：定时器 3 时钟使能(Timer 3 clock enable) 由软件置'1'或清'0' 0：定时器 3 时钟关闭； 1：定时器 3 时钟开启。
位 0	**TIM2EN**：定时器 2 时钟使能(Timer 2 clock enable) 由软件置'1'或清'0' 0：定时器 2 时钟关闭； 1：定时器 2 时钟开启。

7.3.9 备份域控制寄存器 (RCC_BDCR)

偏移地址：0x20

复位值：0x0000 0000，只能由备份域复位有效复位

访问：0 到 3 等待周期，字、半字和字节访问 当连续对该寄存器进行访问时，将插入等待状态。

注意： *备份域控制寄存器中(RCC_BDCR)的LSEON、LSEBYP、RTCSEL 和RTCEN 位处于备份域。因 此，这些位在复位后处于写保护状态，只有在电源控制寄存器(PWR_CR)中的 DBP 位置'1'后才 能对这些位进行改动。进一步信息请参考 7.1 节。这些位只能由备份域复位清除(见 8.1.3 节)。任何内部或外部复位都不会影响这些位。*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															BDRST
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC EN	保留					RTCSEL[1:0]		保留					LSE BYP	LSE RDY	LSEON
rw						rw	rw						rw	r	rw

位 31:17	保留，始终读为 0。
位 16	**BDRST**：备份域软件复位(Backup domain software reset) 由软件置'1'或清'0' 0：复位未激活； 1：复位整个备份域。
位 15	RTCEN：RTC 时钟使能 (RTC clock enable) 由软件置'1'或清'0' 0：RTC 时钟关闭； 1：RTC 时钟开启。
位 14:10	保留，始终读为 0。
位 9:8	**RTCSEL[1:0]**：RTC 时钟源选择(RTC clock source selection) 由软件设置来选择 RTC 时钟源。一旦 RTC 时钟源被选定，直到下次后备域被复位，它不能在被改变。可通过设置 BDRST 位来清除。 00：无时钟； 01：LSE 振荡器作为 RTC 时钟； 10：LSI 振荡器作为 RTC 时钟； 11：HSE 振荡器在 128 分频后作为 RTC 时钟。
位 7:3	保留，始终读为 0。
位 2	**LSEBYP**：外部低速时钟振荡器旁路(External low-speed oscillator bypass) 在调试模式下由软件置'1'或清'0'来旁路 LSE。只有在外部 32kHz 振荡器关闭时，才能写入该位 0：LSE 时钟未被旁路； 1：LSE 时钟被旁路。
位 1	**LSERDY**：外部低速 LSE 就绪(External low-speed oscillator ready) 由硬件置'1'或清'0'来指示是否外部 32kHz 振荡器就绪。在 LSEON 被清零后，该位需要 6 个外部低速振荡器的周期才被清零。 0：外部 32kHz 振荡器未就绪； 1：外部 32kHz 振荡器就绪。
位 0	**LSEON**：外部低速振荡器使能(External low-speed oscillator enable) 由软件置'1'或清'0' 0：外部 32kHz 振荡器关闭； 1：外部 32kHz 振荡器开启。

7.3.10 控制/状态寄存器 (RCC_CSR)

偏移地址：0x24

复位值：0x0C00 0000，除复位标志外由系统复位清除，复位标志只能由电源复位清除。

访问：0 到 3 等待周期，字、半字和字节访问

当连续对该寄存器进行访问时，将插入等待状态。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR RSTF	WWDG RSTF	IWDG RSTF	SFT RSTF	POR RSTF	PIN RSTF	保留	RMVF	保留							
rw	rw	rw	rw	rw	rw		rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														LSI RDY	LSION
														r	rw

位 31	**LPWRRSTF**：低功耗复位标志(Low-power reset flag) 在低功耗管理复位发生时由硬件置'1'；由软件通过写 RMVF 位清除。 0：无低功耗管理复位发生; 1：发生低功耗管理复位. 关于低功耗管理复位的详细信息，请参考 8.1.1 节的"低功耗管理复位"。
位 30	**WWDGRSTF**：窗口看门狗复位标志(Window watchdog reset flag) 在窗口看门狗复位发生时由硬件置' 1' ；由软件通过写 RMVF 位清除。 0：无窗口看门狗复位发生； 1：发生窗口看门狗复位。
位 29	**IWDGRSTF**：独立看门狗复位标志(Independent watchdog reset flag) 在独立看门狗复位发生在 V_{DD} 区域时由硬件置'1'；由软件通过写 RMVF 位清除。 0：无独立看门狗复位发生； 1：发生独立看门狗复位。
位 28	**SFTRSTF**：软件复位标志(Software reset flag) 在软件复位发生时由硬件置'1'；由软件通过写 RMVF 位清除。 0：无软件复位发生； 1：发生软件复位。
位 27	**PORRSTF**：上电/掉电复位标志(POR/PDR reset flag) 在上电/掉电复位发生时由硬件置'1'；由软件通过写 RMVF 位清除。 0：无上电/掉电复位发生； 1：发生上电/掉电复位。
位 26	**PINRSTF**：NRST 引脚复位标志(PIN reset flag) 在 NRST 引脚复位发生时由硬件置'1'；由软件通过写 RMVF 位清除。 0：无 NRST 引脚复位发生； 1：发生 NRST 引脚复位。
位 25	保留，读操作返回 0
位 24	**RMVF**：清除复位标志(Remove reset flag) 由软件置'1'来清除复位标志。 0：无作用； 1：清除复位标志。
位 23:2	保留，读操作返回 0
位 1	**LSIRDY**：内部低速振荡器就绪(Internal low-speed oscillator ready) 由硬件置'1 或清'0'来指示内部 40kHz RC 振荡器是否就绪。在 LSION 清零后，3 个内部 40kHz RC 振荡器的周期后 LSIRDY 被清零。 0：内部 40kHz RC 振荡器时钟未就绪； 1：内部 40kHz RC 振荡器时钟就绪。

位 0	**LSION**：内部低速振荡器使能(Internal low-speed oscillator enable) 由软件置'1 或清'0'。 0：内部 40kHz RC 振荡器关闭； 1：内部 40kHz RC 振荡器开启。

7.3.11 控制寄存器 (RCC_HSECTL)

偏移地址：0x28

复位值：0x 01040004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
							rw							rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		保留					HSEWT HSENF_BYP	保留						XCH_IOP	
		rw					...							rw	

位 31：25	保留
位 24	**HSENF_BYP**： 0 : HSE 时钟信号经过片内滤噪器后被使用。 1 : HSE 时钟信号不经过片内滤噪器。
位 23:19	保留
位 18:16	**XCH_IOP[2:0]**： 000：驱动电流大约为 20uA，推荐使用 0.5 ~ 3MHz 晶振。 001：驱动电流大约为 30uA，推荐使用 0.5 ~ 4MHz 晶振。 010：驱动电流大约为 40uA，推荐使用 0.5 ~ 5MHz 晶振。 011：驱动电流大约为 50uA，推荐使用 1 ~ 6MHz 晶振。 100：驱动电流大约为 60uA，推荐使用 2 ~ 12MHz 晶振。 101：驱动电流大约为 80uA，推荐使用 2 ~ 15MHz 晶振。 110：驱动电流大约为 100uA，推荐使用 3 ~ 20MHz 晶振。 111：驱动电流大约为 200uA，推荐使用 5 ~ 30MHz 晶振。
位 15:14	保留
位 13:0	**HSEWT[13:0]**： HSE 晶振振荡等待时间设置，等待时间 = HSEWT × 8× HSE 时钟周期。 当使用 8MHz 振荡器时，初始等待时间值为：0x100 × 8 × 125nS = 256uS。

7.3.12 控制寄存器 (RCC_LPCLK_CTL)

偏移地址：0x30 复位值：0x 0100

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留							USBCLK_EN							LCLK_SEL	

位 15:9	保留
位 8	**USBCLK_EN：** USBCLK_EN 比特用于在工作模式或者 SLEEP 模式下关闭 USB 48MHz 时钟。 关闭 USB 48MHz 时钟后，MCU 芯片电流功耗可降低约 400uA
位 7:1	保留
位 0	**LCLK_SEL[1:0]:** 0：选择 LSI 作为系统时钟 1：选择 LSE 作为系统时钟

8 通用和复用功能 I/O(GPIO 和 AFIO)

8.1 GPIO 功能描述

每个 GPI/O 端口有两个 32 位配置寄存器(GPIOx_CRL，GPIOx_CRH)，两个 32 位数据寄存器(GPIOx_IDR 和 GPIOx_ODR)，一个 32 位置位/复位寄存器(GPIOx_BSRR)，一个 16 位复位寄存 器(GPIOx_BRR)和一个 32 位锁定寄存器(GPIOx_LCKR)。

根据数据手册中列出的每个 I/O 端口的特定硬件特征，GPIO 端口的每个位可以由软件分别配置成多种模式。

- 输入浮空
- 输入上拉
- 输入下拉
- 模拟输入
- 开漏输出
- 推挽式输出
- 推挽式复用功能
- 开漏复用功能

每个 I/O 端口位可以自由编程，然而必须按照 32 位字访问 I/O 端口寄存器(不允许半字或字节访问)。GPIOx_BSRR 和 GPIOx_BRR 寄存器允许对任何 GPIO 寄存器进行读/更改的独立访问；这样，在读和更改访问之间产生 IRQ 时不会发生危险。

下图给出了一个 I/O 端口位的基本结构。

图 9 I/O 端口位的基本结构

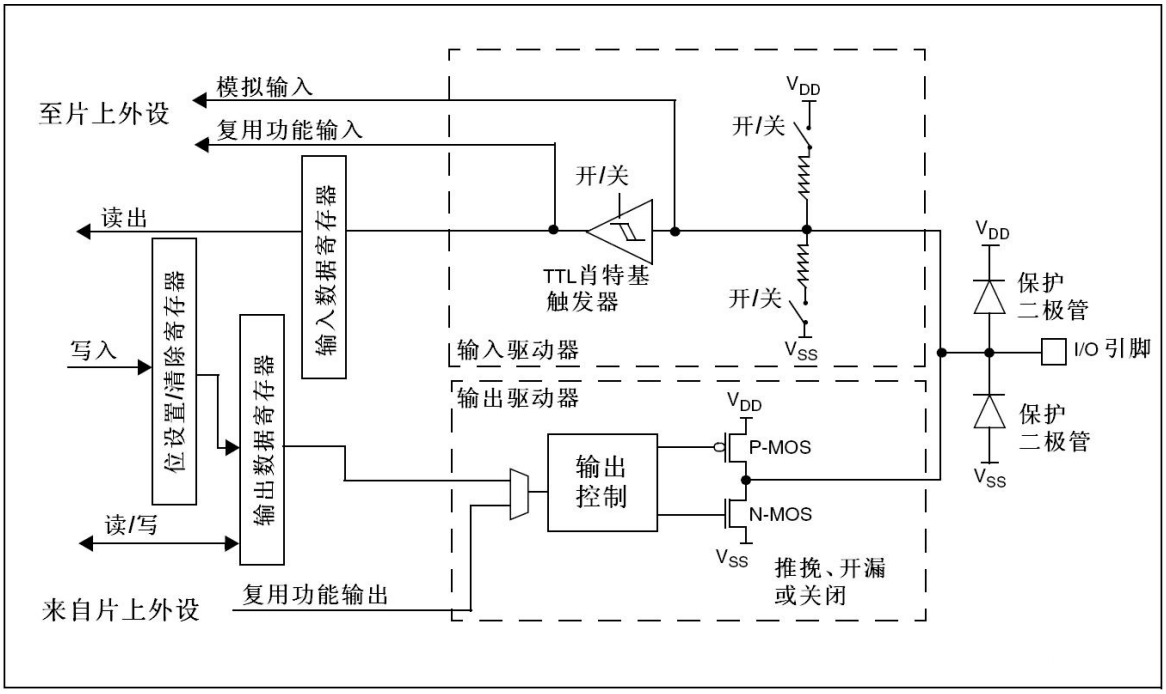
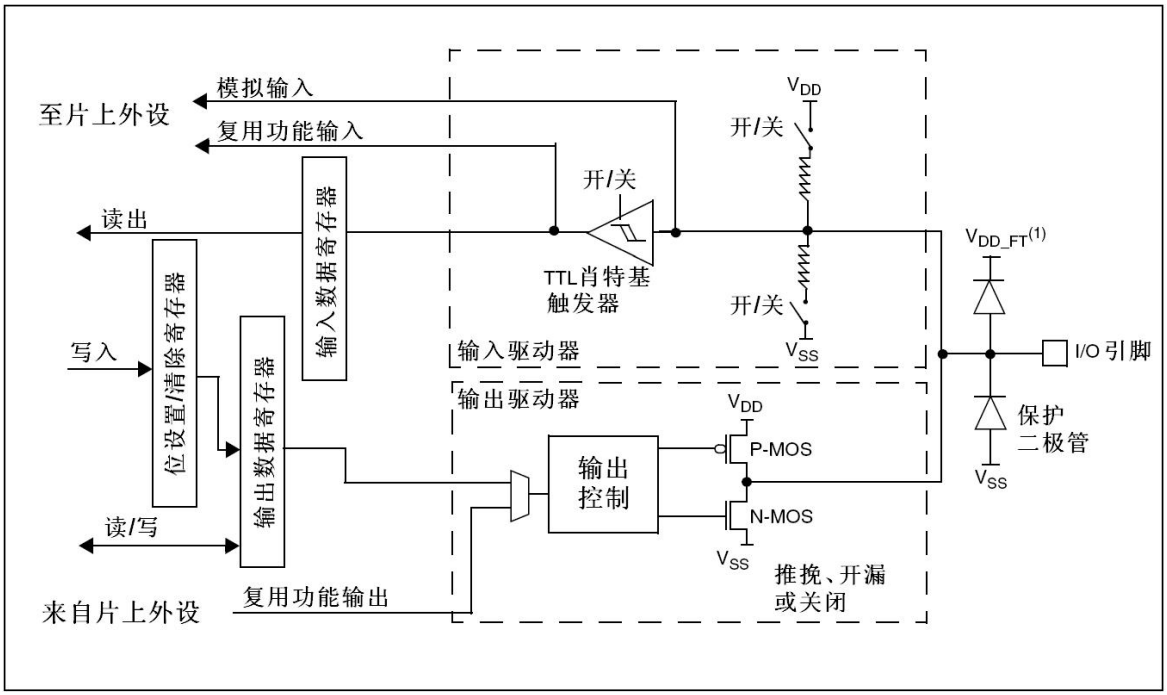


图 10 5 伏兼容 I/O 端口位的基本结构



(1) V_{DD_FT} 对 5 伏容忍 I/O 脚是特殊的，它与 V_{DD} 不同

表 13 端口位配置表

配置模式		CNF1	CNF0	MODE1	MODE0	PxODR 寄存器
通用输出	推挽(Push-Pull)	0	0	01 10 11 见表 14		0 或1
	开漏(Open-Drain)		1			0 或1
复用功能输出	推挽(Push-Pull)	1	0			不使用
	开漏(Open-Drain)		1			不使用
输入	模拟输入	0	0	00		不使用
	浮空输入		1			不使用
	下拉输入	1	0			0
	上拉输入					1

表 14 输出模式位

MODE[1:0]	意义
00	保留
01	最大输出速度为 10MHz
10	最大输出速度为 2MHz
11	最大输出速度为 50MHz

8.1.1 通用 I/O(GPIO)

复位期间和刚复位后，复用功能未开启，I/O 端口被配置成浮空输入模式(CNFx[1:0]=01b，MODEx[1:0]=00b)。

复位后，JTAG 引脚被置于输入上拉或下拉模式：

- PA15：JTDI 置于上拉模式
- PA14：JTCK 置于下拉模式
- PA13：JTMS 置于上拉模式
- PB4： JNTRST 置于上拉模式

当作为输出配置时，写到输出数据寄存器上的值(GPIOx_ODR)输出到相应的 I/O 引脚。可以以推挽模式或开漏模式(当输出 0 时，只有 N-MOS 被打开)使用输出驱动器。

输入数据寄存器(GPIOx_IDR)在每个 APB2 时钟周期捕捉 I/O 引脚上的数据。所有 GPIO 引脚有一个内部弱上拉和弱下拉，当配置为输入时，它们可以被激活也可以被断开。

8.1.2 单独的位设置或位清除

当对 GPIOx_ODR 的个别位编程时，软件不需要禁止中断：在单次 APB2 写操作里，可以只更改 一个或多个位。

这是通过对“置位/复位寄存器”(GPIOx_BSRR，复位是 GPIOx_BRR)中想要更改的位写'1'来实现的。没被选择的位将不被更改。

8.1.3 外部中断/唤醒线

所有端口都有外部中断能力。为了使用外部中断线，端口必须配置成输入模式。更多的关于外部中断的信息，参考：

- 第 9.2 节：外部中断/事件控制器(EXTI)；
- 第 9.2.3 节：唤醒事件管理。

8.1.4 复用功能(AF)

使用默认复用功能前必须对端口位配置寄存器编程。

- 对于复用的输入功能，端口必须配置成输入模式(浮空、上拉或下拉)且输入引脚必须由外部驱动

注意：也可以通过软件来模拟复用功能输入引脚，这种模拟可以通过对 GPIO 控制器编程来实现。此时，端口应当被设置为复用功能输出模式。显然，这时相应的引脚不再由外部驱动，而是通过 GPIO 控制器由软件来驱动。

- 对于复用输出功能，端口必须配置成复用功能输出模式(推挽或开漏)。
- 对于双向复用功能，端口位必须配置复用功能输出模式(推挽或开漏)。这时，输入驱动器被配置成浮空输入模式。

如果把端口配置成复用输出功能，则引脚和输出寄存器断开，并和片上外设的输出信号连接。

如果软件把一个 GPIO 脚配置成复用输出功能，但是外设没有被激活，它的输出将不确定。

8.1.5 软件重新映射 I/O 复用功能

为了使不同器件封装的外设 I/O 功能的数量达到最优，可以把一些复用功能重新映射到其他一些脚上。这可以通过软件配置相应的寄存器来完成(参考 AFIO 寄存器描述)。这时，复用功能就不再映射到它们的原始引脚上了。

8.1.6 GPIO 锁定机制

锁定机制允许冻结 IO 配置。当在一个端口位上执行了锁定(LOCK)程序，在下一次复位之前，将不能再更改端口位的配置。

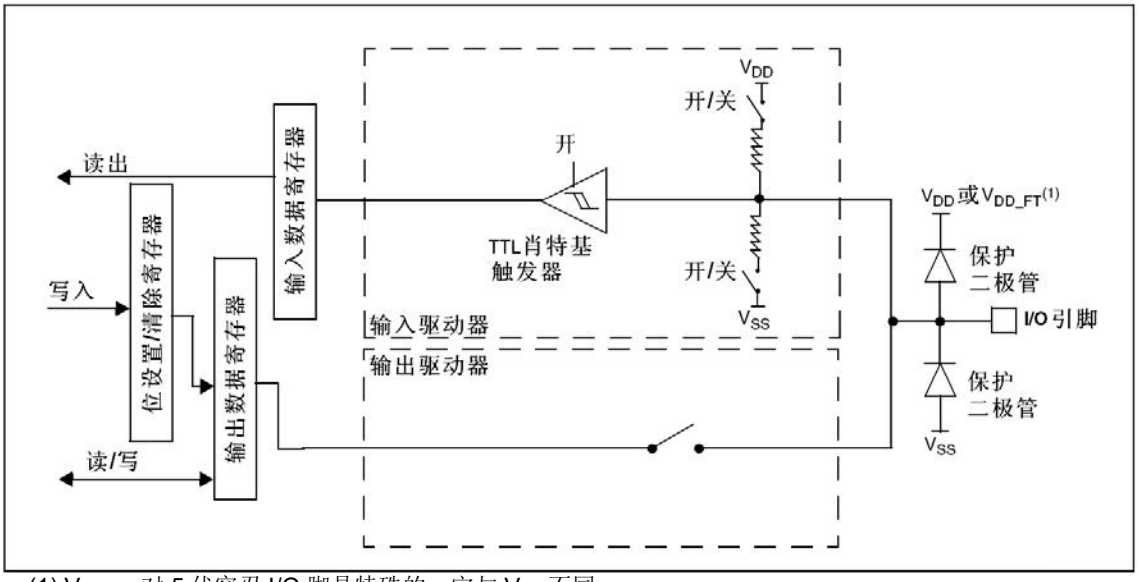
8.1.7 输入配置

当 I/O 端口配置为输入时：

- 输出缓冲器被禁止
- 施密特触发输入被激活
- 根据输入配置(上拉，下拉或浮动)的不同，弱上拉和下拉电阻被连接
- 出现在 I/O 脚上的数据在每个 APB2 时钟被采样到输入数据寄存器
- 对输入数据寄存器的读访问可得到 I/O 状态

下图给出了 I/O 端口位的输入配置

图 11 输入浮空/上拉/下拉配置



(1) V_{DD_FT} 对 5 伏容忍 I/O 脚是特殊的，它与 V_{DD} 不同

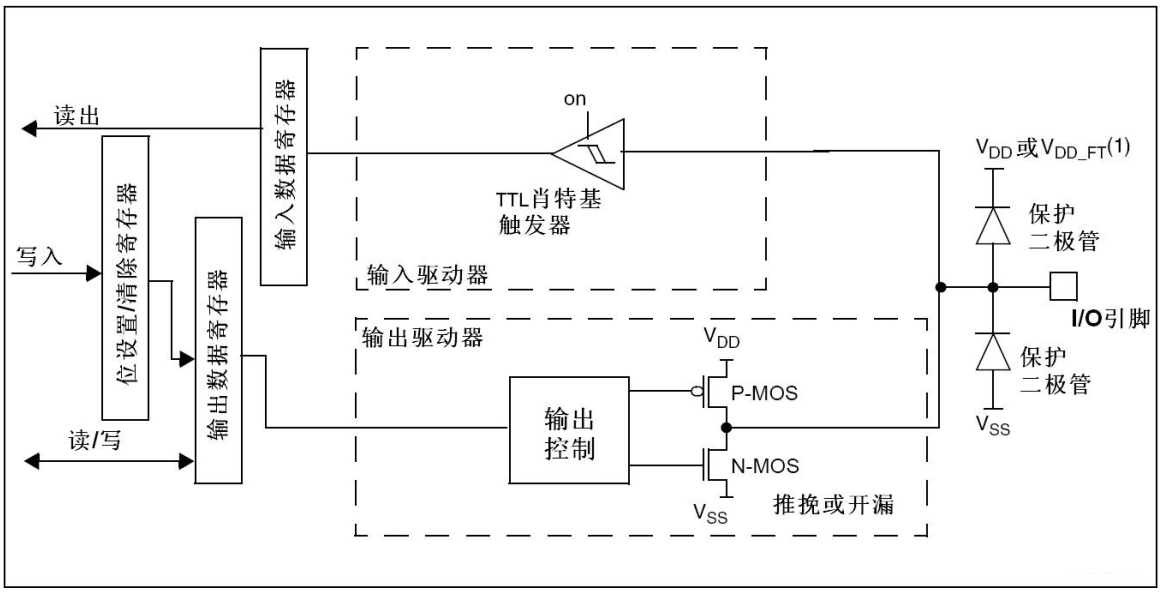
8.1.8 输出配置

当 I/O 端口被配置为输出时：

- 输出缓冲器被激活
  - 开漏模式：输出寄存器上的’0’激活 N-MOS，而输出寄存器上的’1’将端口置于高阻状态(P-MOS 从不被激活)。
  - 推挽模式：输出寄存器上的’0’激活 N-MOS，而输出寄存器上的’1’将激活 P-MOS。
- 施密特触发输入被激活
- 弱上拉和下拉电阻被禁止
- 出现在 I/O 脚上的数据在每个 APB2 时钟被采样到输入数据寄存器
- 在开漏模式时，对输入数据寄存器的读访问可得到 I/O 状态
- 在推挽式模式时，对输出数据寄存器的读访问得到最后一次写的值。

下图给出了 I/O 端口位的输出配置。

图 12 输出配置



(1) V_{DD_FT} 对5 伏兼容I/O 脚是特殊的，它与V_{DD} 不同

8.1.9 复用功能配置

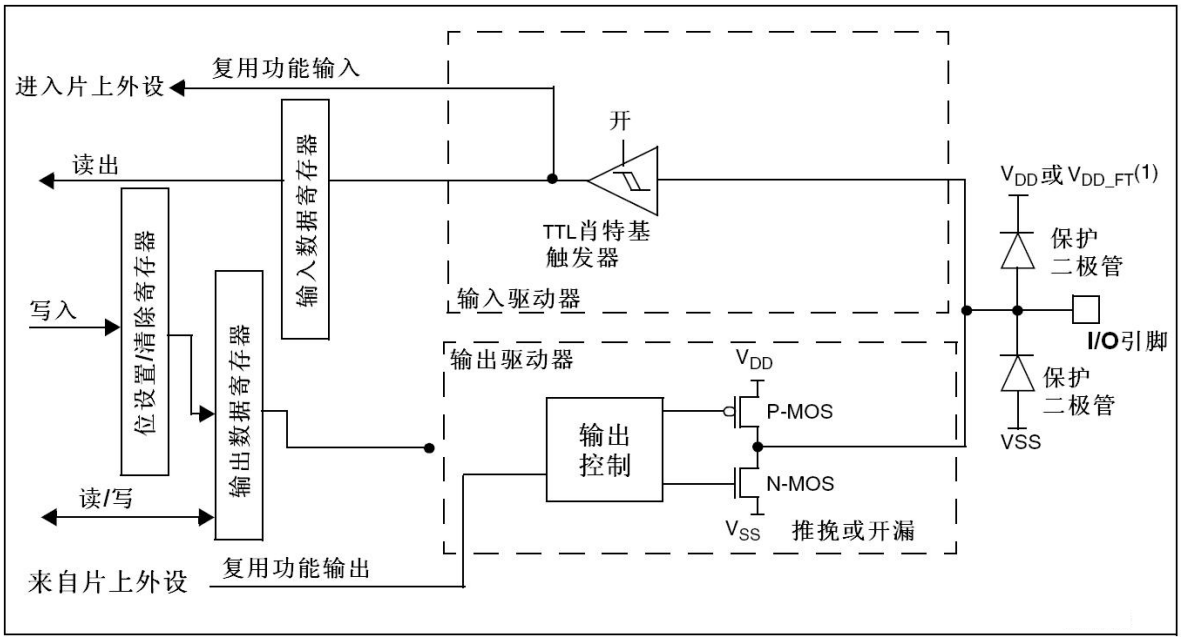
当 I/O 端口被配置为复用功能时：

- 在开漏或推挽式配置中，输出缓冲器被打开
- 内置外设的信号驱动输出缓冲器(复用功能输出)
- 施密特触发输入被激活
- 弱上拉和下拉电阻被禁止
- 在每个 APB2 时钟周期，出现在 I/O 脚上的数据被采样到输入数据寄存器
- 开漏模式时，读输入数据寄存器时可得到 I/O 口状态
- 在推挽模式时，读输出数据寄存器时可得到最后一次写的值

下图示出了 I/O 端口位的复用功能配置。详见 9.4 节-AFIO 寄存器描述。

一组复用功能 I/O 寄存器允许用户把一些复用功能重新映象到不同的引脚。

图 13 复用功能配置



(1) V_{DD_FT} 对 5 伏兼容 I/O 脚是特殊的，它与 V_{DD} 不同

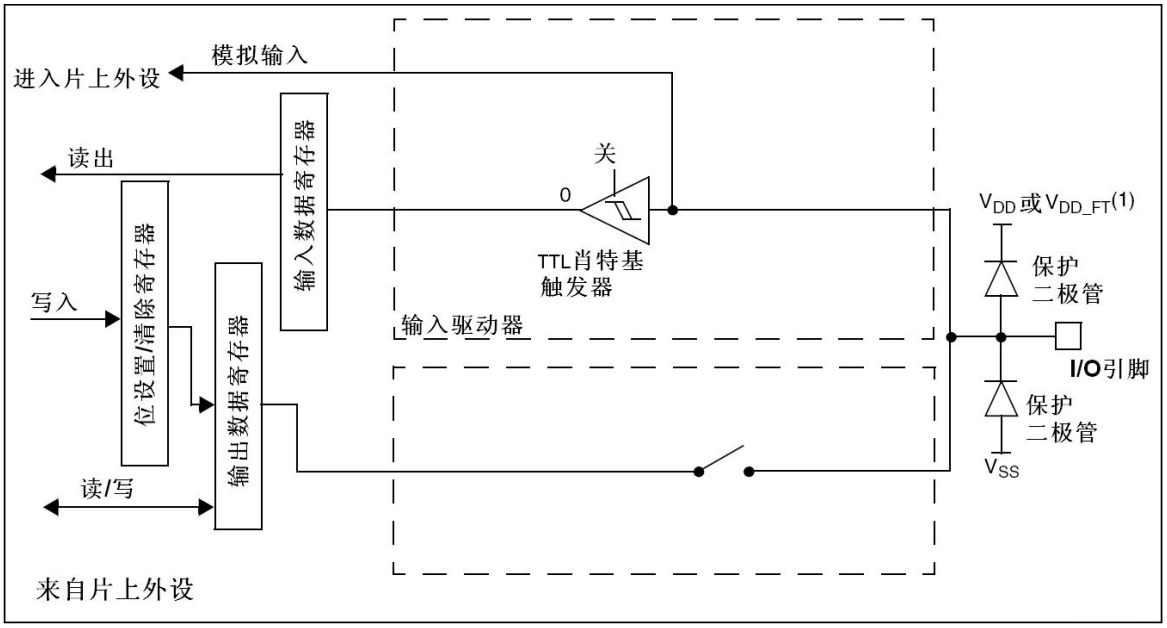
8.1.10 模拟输入配置

当 I/O 端口被配置为模拟输入配置时：

- 输出缓冲器被禁止；
- 禁止施密特触发输入，实现了每个模拟 I/O 引脚上的零消耗。施密特触发输出值被强置为'0'；
- 弱上拉和下拉电阻被禁止；
- 读取输入数据寄存器时数值为'0'。

下图示出了 I/O 端口位的高阻抗模拟输入配置：

图 14 高阻抗的模拟输入配置



(1) V_{DD_FT} 对 5 伏兼容 I/O 脚是特殊的，它与 V_{DD} 不同

8.1.11 外设的 GPIO 配置

下列表格列出了各个外设的引脚配置。

表 15 高级定时器 TIM1

TIM1 引脚	配置	GPIO 配置
TIM1/8_CHx	输入捕获通道 x	浮空输入
	输出比较通道 x	推挽复用输出
TIM1/8_CHxN	互补输出通道 x	推挽复用输出
TIM1/8_BKIN	刹车输入	浮空输入
TIM1/8_ETR	外部触发时钟输入	浮空输入

表 16 通用定时器 TIM2/3/4/5

TIM2/3/4/5 引脚	配置	GPIO 配置
TIM2/3/4/5_CHx	输入捕获通道 x	浮空输入
	输出比较通道 x	推挽复用输出
TIM2/3/4/5_ETR	外部触发时钟输入	浮空输入

表 17 USART

USART 引脚	配置	GPIO 配置
USARTx_TX(1)	全双工模式	推挽复用输出
	半双工同步模式	推挽复用输出
USARTx_RX	全双工模式	浮空输入或带上拉输入
	半双工同步模式	未用，可作为通用 I/O
USARTx_CK	同步模式	推挽复用输出
USARTx_RTS	硬件流量控制	推挽复用输出
USARTx_CTS	硬件流量控制	浮空输入或带上拉输入

(1):USART_TX 也可以配置为复用开漏模式。

表 18 SPI

SPI 引脚	配置	GPIO 配置
SPIx_SCK	主模式	推挽复用输出
	从模式	浮空输入
SPIx_MOSI	全双工模式/主模式	推挽复用输出
	全双工模式/从模式	浮空输入或带上拉输入
	简单的双向数据线/主模式	推挽复用输出
	简单的双向数据线/从模式	未用，可作为通用 I/O
SPIx_MISO	全双工模式/主模式	浮空输入或带上拉输入
	全双工模式/从模式	推挽复用输出
	简单的双向数据线/主模式	未用，可作为通用 I/O
	简单的双向数据线/从模式	推挽复用输出
SPIx_NSS	硬件主/从模式	浮空输入或带上拉输入或带下拉输入
	硬件主模式/NSS 输出使能	推挽复用输出
	软件模式	未用，可作为通用 I/O

表 19 I2C 接口

I2C 引脚	配置	GPIO 配置
I2Cx_SCL	I2C 时钟	开漏复用输出
I2Cx_SDA	I2C 数据	开漏复用输出

表 20 BxCAN

BxCAN 引脚	GPIO 配置
CAN_TX	推挽复用输出
CAN_RX	浮空输入或带上拉输入

表 21 USB

USB 引脚	GPIO 配置
USB_DM / USB_DP	一旦使能了 USB 模块，这些引脚会自动连接到内部 USB 收发器

ADC 输入引脚必须配置为模拟输入

表 22 ADC/DAC

ADC/DAC 引脚	GPIO 配置
ADC/DAC	模拟输入

表 23 其它 I/O 功能

引脚	复用功能	GPIO 配置
TAMPER-RTC	RTC 输出	当配置 BKP_CR 和BKP_RTCCR 寄存器时，由硬件强制设置
	侵入事件输入	
MCO	时钟输出	推挽复用输出
EXTI 输入线	外部中断输入	浮空输入或带上拉输入或带下拉输入

8.2 GPIO 寄存器描述

请参考第 2 章中有关寄存器描述中用到的缩写。必须以字(32 位)的方式操作这些外设寄存器。

8.2.1 端口配置低寄存器(GPIOx_CRL) (x=A..D)

偏移地址：0x00

复位值：0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30 27:26 23:22 19:18 15:14 11:10 7:6 3:2	**CNFy[1:0]**：端口 x 配置位(y = 0…7) (Port x configuration bits) 软件通过这些位配置相应的 I/O 端口，请参考表 13 端口位配置表。 在输入模式(MODE[1:0]=00)： 00：模拟输入模式 01：浮空输入模式(复位后的状态) 10：上拉/下拉输入模式 11：保留 在输出模式 (MODE[1:0]>00)： 00：通用推挽输出模式 01：通用开漏输出模式 10：复用功能推挽输出模式 11：复用功能开漏输出模式
位 29:28 25:24 21:20 17:16 13:12 9:8, 5:4 1:0	**MODEy[1:0]**：端口 x 的模式位(y = 0…7) (Port x mode bits) 软件通过这些位配置相应的 I/O 端口，请参考表 13 端口位配置表。 00：输入模式(复位后的状态) 01：输出模式，最大速度 10MHz 10：输出模式，最大速度 2MHz 11：输出模式，最大速度 50MHz

8.2.2 端口配置高寄存器(GPIOx_CRH) (x=A..D)

偏移地址：0x04

复位值：0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30 27:26 23:22 19:18 15:14 11:10 7:6 3:2	**CNFy[1:0]**：端口 x 配置位(y = 8…15) (Port x configuration bits) 软件通过这些位配置相应的 I/O 端口，请参考表 13 端口位配置表。 在输入模式(MODE[1:0]=00)： 00：模拟输入模式 01：浮空输入模式(复位后的状态) 10：上拉/下拉输入模式 11：保留 在输出模式(MODE[1:0]>00)： 00：通用推挽输出模式 01：通用开漏输出模式 10：复用功能推挽输出模式 11：复用功能开漏输出模式
位 9:28 25:24 21:20 17:16 13:12 9:8, 5:4 1:0	**MODEy[1:0]**：端口 x 的模式位(y = 8…15) (Port x mode bits) 软件通过这些位配置相应的 I/O 端口，请参考表 13 端口位配置表。 00：输入模式(复位后的状态) 01：输出模式，最大速度 10MHz 10：输出模式，最大速度 2MHz 11：输出模式，最大速度 50MHz

8.2.3 端口输入数据寄存器(GPIOx_IDR) (x=A..D)

地址偏移：0x08

复位值：0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:16	保留，始终读为 0。
位 15:0	**IDRy[15:0]**：端口输入数据(y = 0…15) (Port input data) 这些位为只读并只能以字(16 位)的形式读出。读出的值为对应 I/O 口的状态。

8.2.4 端口输出数据寄存器(GPIOx_ODR) (x=A..D)

地址偏移：0Ch

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16	保留，始终读为 0。
位 15:0	**ODRy[15:0]**：端口输出数据(y = 0…15) (Port output data) 这些位可读可写并只能以字(16 位)的形式操作。 注：对 GPIOx_BSRR(x = A…D)，可以分别地对各个 ODR 位进行独立的设置/清除。

8.2.5 端口位设置/清除寄存器(GPIOx_BSRR) (x=A..D)

地址偏移：0x10

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 31:16	**BRy**: 清除端口 x 的位 y (y = 0…15) (Port x Reset bit y) 这些位只能写入并只能以字(16 位)的形式操作。 0：对对应的 ODRy 位不产生影响 1：清除对应的 ODRy 位为 0 注：如果同时设置了 BSy 和 BRy 的对应位，BSy 位起作用。
位 15:0	**BSy**: 设置端口 x 的位 y (y = 0…15) (Port x Set bit y) 这些位只能写入并只能以字(16 位)的形式操作。 0：对对应的 ODRy 位不产生影响 1：设置对应的 ODRy 位为 1

8.2.6 端口位清除寄存器(GPIOx_BRR) (x=A..D)

地址偏移：0x14

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 31:16	保留。
位 15:0	**BRy**: 清除端口 x 的位 y (y = 0…15) (Port x Reset bit y) 这些位只能写入并只能以字(16 位)的形式操作。 0：对对应的 ODRy 位不产生影响 1：清除对应的 ODRy 位为 0

8.2.7 端口配置锁定寄存器(GPIOx_LCKR) (x=A..D)

当执行正确的写序列设置了位 16(LCKK)时，该寄存器用来锁定端口位的配置。位[15:0]用于锁 定 GPIO 端口的配置。在规定的写入操作期间，不能改变 LCKP[15:0]。当对相应的端口位执行了LOCK 序列后，在下次系统复位之前将不能再更改端口位的配置。

每个锁定位锁定控制寄存器(CRL, CRH)中相应的 4 个位。

地址偏移：0x18

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															LCKK
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:17	保留。
位 16	LCKK：锁键(Lock key) 该位可随时读出，它只可通过锁键写入序列修改。 0：端口配置锁键位激活 1：端口配置锁键位被激活，下次系统复位前 GPIOx_LCKR 寄存器被锁住。 锁键的写入序列： 写 1 -> 写 0 -> 写 1 -> 读 0 -> 读 1 最后一个读可省略，但可以用来确认锁键已被激活。 注：在操作锁键的写入序列时，不能改变 LCK[15:0]的值。 操作锁键写入序列中的任何错误将不能激活锁键。
位 15:0	LCKy: 端口 x 的锁位 y (y = 0…15) (Port x Lock bit y) 这些位可读可写但只能在 LCKK 位为 0 时写入。 0：不锁定端口的配置 1：锁定端口的配置

8.3 复用功能 I/O 和调试配置(AFIO)

为了优化 64 脚封装的外设数目，可以把一些复用功能重新映射到其他引脚上。设置复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)实现引脚的重新映射。这时，复用功能不再映射到它们的原始分配上。

8.3.1 把 OSC32_IN/OSC32_OUT 作为 GPIO 端口 PC14/PC15

当LSE 振荡器关闭时，LSE 振荡器引脚 OSC32_IN/OSC32_OUT 可以分别用做 GPIO 的 PC14/PC15，LSE 功能始终优先于通用 I/O 口的功能。

注： *1. 当关闭 1.8V 电压区(进入待机模式)或后备区域使用 V_{BAT} 供电(不再有 V_{DD} 供电)时，不能使用 PC14/PC15 的 GPIO 口功能；*

2. 参见第 6.1.2 节有关 I/O 口使用的限制

8.3.2 把 OSC_IN/OSC_OUT 引脚作为 GPIO 端口 PD0/PD1

外部振荡器引脚 OSC_IN/OSC_OUT 可以用做 GPIO 的 PD0/PD1，通过设置复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)实现。

注： *外部中断/事件功能没有被重映射。在 48 和 64 脚的封装上，PD0 和 PD1 不能用来产生外部中断/事件。*

8.3.3 CAN 复用功能重映射

CAN 信号可以被映射到端口 A、端口 B 上，如下表所示。。

表 24 CAN 复用功能重映射

复用功能(1)	CAN_REMAP[1:0]="00"	CAN_REMAP[1:0]="10"
CAN_RX 或AN_RX	PA11	PB8
CAN_TX 或AN_TX	PA12	PB9

1. CAN_RX 和 CAN_TX。

8.3.4 JTAG/SWD 复用功能重映射

调试接口信号被映射到 GPIO 端口上，如下表所示。

表 25 调试接口信号

复用功能	GPIO 端口
JTMS/SWDIO	PA13
JTCK/SWCLK	PA14
JTDI	PA15
JTDO/TRACESWO	PB3
JNTRST	PB4
TRACECK	PE2
TRACED0	PE3
TRACED1	PE4
TRACED2	PE5
TRACED3	PE6

为了在调试期间可以使用更多 GPIOs，通过设置复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)的 SWJ_CFG[2:0]位，可以改变上述重映像配置。参见下表。

表 26 调试端口映像

SWJ_CFG [2:0]	可能的调试端口	SWJ I/O 引脚分配				
		PA13/ JTMS/ SWDIO	PA14/ JTCK/ SWCLK	PA15/ JTDI	PB3/ JTDO/ TRACESWO	PB4/ NJTRST
000	完全 SWJ(JTAG-DP + SW-DP) (复位状态)	I/O 不可用	I/O 不可用	I/O 不可用	I/O 不可用	I/O 不可用
001	完全 SWJ(JTAG-DP + SW-DP) 但没有 JNTRST	I/O 不可用	I/O 不可用	I/O 不可用	I/O 不可用	I/O 可用
010	关闭 JTAG-DP， 启用 SW-DP	I/O 不可用	I/O 不可用	I/O 可用	I/O 可用(1)	I/O 可用
100	关闭 JTAG-DP， 关闭 SW-DP	I/O 可用	I/O 可用	I/O 可用	I/O 可用	I/O 可用
其它	禁用					

1. I/O 口只可在不使用异步跟踪时使用。

8.3.5 定时器复用功能重映射

定时器 4 的通道 1 到通道 4 可以从端口 B 重映射到端口 D。其他定时器的重映射列在表 28~表 30。参见复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)。

表 27 TIM4 复用功能重映像

复用功能	TIM4_REMAP = 0
TIM4_CH1	PB6
TIM4_CH2	PB7
TIM4_CH3	PB8
TIM4_CH4	PB9

表 28 TIM3 复用功能重映像

复用功能	TIM3_REMAP[1:0] = 00 (没有重映像)	TIM3_REMAP[1:0] = 10 (部分重映像)	TIM3_REMAP[1:0] = 11 (完全重映像)(1)
TIM3_CH1	PA6	PB4	PC6
TIM3_CH2	PA7	PB5	PC7
TIM3_CH3	PB0		PC8
TIM3_CH4	PB1		PC9

1. 重映像只适用于64 脚的封装

表 29 TIM2 复用功能重映像

复用功能	TIM2_REMAP[1:0] =00 (没有重映像)	TIM2_REMAP[1:0] =01 (部分重映像)	TIM2_REMAP[1:0] =10 (部分重映像)(1)	TIM2_REMAP[1:0] =11 (完全重映像)
TIM2_CH1_ETR(1)	PA0	PA15	PA0	PA15
TIM2_CH2	PA1	PB3	PA1	PB3
TIM2_CH3	PA2		PB10	
TIM2_CH4	PA3		PB11	

1. TIM2_CH1 **和** TIM2_ETR **共用一个引脚，但不能同时使用(因此在此使用这样的标记：** TIM2_CH1_ETR)

表 30 TIM1 复用功能重映像

复用功能映像	TIM1_REMAP[1:0] = 00 (没有重映像)	TIM1_REMAP[1:0] = 01 (部分重映像)
TIM1_ETR	PA12	
TIM1_CH1	PA8	
TIM1_CH2	PA9	
TIM1_CH3	PA10	
TIM1_CH4	PA11	
TIM1_BKIN	保留	PA6
TIM1_CH1N	保留	PA7
TIM1_CH2N	保留	PB0
TIM1_CH3N	保留	PB1

8.3.6 USART 复用功能重映射

参见复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)

表 31 USART3 重映像

复用功能	USART3_REMAP[1:0] = 00 (没有重映像)	USART3_REMAP[1:0] = 01 (部分重映像)(1)	USART3_REMAP[1:0] = 11 (完全重映像)(2)
USART3_TX	PB10	PC10	保留
USART3_RX	PB11	PC11	保留
USART3_CK	PB12	PC12	保留
USART3_CTS	PB13		保留
USART3_RTS	PB14		保留

1. 重映像只适用于 64 脚的封装

表 32 USART1 重映像

复用功能	USART1_REMAP = 0	USART1_REMAP = 1
USART1_TX	PA9	PB6
USART1_RX	PA10	PB7

8.3.7 I2C1 复用功能重映射

参见复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)。

表 33 I2C 1 重映像

复用功能	I2C1_REMAP = 0	I2C1_REMAP = 1
I2C1_SCL	PB6	PB8
I2C1_SDA	PB7	PB9

8.3.8 SPI 1 复用功能重映射

参见复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)。

表 34 SPI1 重映像

复用功能	SPI1_REMAP = 0	SPI1_REMAP = 1
SPI1_NSS	PA4	PA15
SPI1_SCK	PA5	PB3
SPI1_MISO	PA6	PB4
SPI1_MOSI	PA7	PB5

8.4 AFIO 寄存器描述

请参考第 2 章中有关寄存器描述中用到的缩写。

注意： *对寄存器 AFIO_EVCR，AFIO_MAPR 和 AFIO_EXTICRX 进行读写操作前，应当首先打开 AFIO 的时钟。参考第 8.3.7 节 APB2 外设时钟使能寄存器(RCC_APB2ENR)。*
必须以字(32 位)的方式操作这些外设寄存器。

8.4.1 事件控制寄存器(AFIO_EVCR)

地址偏移：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								EVOE	PORT[2:0]			PIN[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

位 31:8	保留。
位 7	**EVOE**：允许事件输出(Event output enable) 该位可由软件读写。当设置该位后，Cortex 的 EVENTOUT 将连接到由 PORT[2:0]和 PIN[3:0]选定的 I/O 口。
位 6:4	**PORT[2:0]**：端口选择(Port selection) 选择用于输出 Cortex 的 EVENTOUT 信号的端口： 000：选择 PA　001：选择 PB　010：选择 PC　011：选择 PD 100：选择 PE
位 3:0	**PIN[3:0]**：引脚选择(x=A…D) (Pin selection) 选择用于输出 Cortex 的 EVENTOUT 信号的引脚： 0000：选择 Px0　0001：选择 Px1　0010：选择 Px2　0011：选择 Px3 0100：选择 Px4　0101：选择 Px5　0110：选择 Px6　0111：选择 Px7 1000：选择 Px8　1001：选择 Px9　1010：选择 Px10　1011：选择 Px11 1100：选择 Px12　1101：选择 Px13　1110：选择 Px14　1111：选择 Px15

8.4.2 复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)

地址偏移：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留					SWJ_CFG[2:0]			保留							
					w	w	w								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD01_REMAP	CAN_REMAP[1:0]		TIM4_REMAP	TIM3_REMAP[1:0]		TIM2_REMAP[1:0]		TIM1_REMAP[1:0]		USART3_REMAP[1:0]		USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:27	保留。
位 26:24	**SWJ_CFG[2:0]**：串行线 JTAG 配置(Serial wire JTAG configuration) 这些位只可由软件写(读这些位，将返回未定义的数值)，用于配置 SWJ 和跟踪复用功能的 I/O 口。SWJ(串行线 JTAG)支持 JTAG 或 SWD 访问 Cortex 的调试端口。系统复位后的默认状态是启用 SWJ 但没有跟踪功能，这种状态下可以通过 JTMS/JTCK 脚上的特定信号选择 JTAG 或 SW(串 行线)模式。 000：完全 SWJ(JTAG-DP + SW-DP)：复位状态； 001：完全 SWJ(JTAG-DP + SW-DP)但没有 NJTRST； 010：关闭 JTAG-DP，启用 SW-DP； 100：关闭 JTAG-DP，关闭 SW-DP； 其它组合：无作用。
位 23:21	保留。
位 20	保留。
位 19	保留。
位 18	保留。
位 17	保留。
位 16	保留。
位 15	**PD01_REMAP** ：端口 D0/端口 D1 映像到 OSC_IN/OSC_OUT (Port D0/Port D1 mapping on OSC_IN/OSC_OUT) 该位可由软件置'1'或置'0'。它控制 PD0 和 PD1 的 GPIO 功能映像。当不使用主振荡器 HSE 时(系统运行于内部的 8MHz 阻容振荡器)，PD0 和 PD1 可以映像到 OSC_IN 和 OSC_OUT 引脚。 0：不进行 PD0 和 PD1 的重映像； 1：PD0 映像到 OSC_IN，PD1 映像到 OSC_OUT。

位 14:13	**CAN_REMAP[1:0]**：CAN 复用功能重映像(CAN alternate function remapping) 这些位可由软件置'1'或置'0'，在只有单个 CAN 接口的产品上控制复用功能 CAN_RX 和 CAN_TX 的重映像。 00：CAN_RX 映像到 PA11，CAN_TX 映像到 PA12； 01：未用组合； 10：CAN_RX 映像到 PB8，CAN_TX 映像到 PB9； 11：CAN_RX 映像到 PD0，CAN_TX 映像到 PD1。
位 12	**TIM4_REMAP**：定时器 4 的重映像(TIM4 remapping) 该位可由软件置'1'或置'0'，控制将 TIM4 的通道 1-4 映射到 GPIO 端口上。 0：没有重映像(TIM4_CH1/PB6，TIM4_CH2/PB7，TIM4_CH3/PB8，TIM4_CH4/PB9)； 1：完全映像(TIM4_CH1/PD12，TIM4_CH2/PD13，TIM4_CH3/PD14，TIM4_CH4/PD15)。 注：重映像不影响在 PE0 上的 TIM4_ETR。
位 11:10	**TIM3_REMAP[1:0]**：定时器 3 的重映像(TIM3 remapping) 这些位可由软件置'1'或置'0'，控制定时器 3 的通道 1 至 4 在 GPIO 端口的映像。 00：没有重映像(CH1/PA6，CH2/PA7，CH3/PB0，CH4/PB1)； 01：未用组合； 10：部分映像(CH1/PB4，CH2/PB5，CH3/PB0，CH4/PB1)； 11：完全映像(CH1/PC6，CH2/PC7，CH3/PC8，CH4/PC9)。注： 重映像不影响在 PD2 上的 TIM3_ETR。
位 9:8	**TIM2_REMAP[1:0]**：定时器 2 的重映像(TIM2 remapping) 这些位可由软件置'1'或置'0'，控制定时器 2 的通道 1 至 4 和外部触发(ETR)在 GPIO 端口的映像。 00：没有重映像(CH1/ETR/PA0，CH2/PA1，CH3/PA2，CH4/PA3)； 01：部分映像(CH1/ETR/PA15，CH2/PB3，CH3/PA2，CH4/PA3)； 10：部分映像(CH1/ETR/PA0，CH2/PA1，CH3/PB10，CH4/PB11)； 11：完全映像(CH1/ETR/PA15，CH2/PB3，CH3/PB10，CH4/PB11)。
位 7:6	**TIM1_REMAP[1:0]**：定时器 1 的重映像(TIM1 remapping) 这些位可由软件置'1'或置'0'，控制定时器 1 的通道 1 至 4、1N 至 3N、外部触发(ETR)和刹车输入 (BKIN)在 GPIO 端口的映像。 00：没有重映像(ETR/PA12，CH1/PA8，CH2/PA9，CH3/PA10，CH4/PA11，BKIN/PB12，CH1N/PB13，CH2N/PB14，CH3N/PB15)； 01：部分映像(ETR/PA12，CH1/PA8，CH2/PA9，CH3/PA10，CH4/PA11，BKIN/PA6，CH1N/PA7，CH2N/PB0，CH3N/PB1)； 10: 未用组合； 11: 完全映像(ETR/PE7，CH1/PE9，CH2/PE11，CH3/PE13，CH4/PE14，BKIN/PE15，CH1N/PE8，CH2N/PE10，CH3N/PE12)。
位 5:4	**USART3_REMAP[1:0]**：USART3 的重映像(USART3 remapping) 这些位可由软件置'1'或置'0'，控制 USART3 的 CTS、RTS、CK、TX 和 RX 复用功能在 GPIO 端口的映像。 00: 没有重映像(TX/PB10，RX/PB11，CK/PB12，CTS/PB13，RTS/PB14)； 01: 部分映像(TX/PC10，RX/PC11，CK/PC12，CTS/PB13，RTS/PB14)； 10: 未用组合； 11: 完全映像(TX/PD8，RX/PD9，CK/PD10，CTS/PD11，RTS/PD12)。
位 3	**USART2_REMAP**：USART2 的重映像(USART2 remapping) 这些位可由软件置'1'或置'0'，控制 USART2 的 CTS、RTS、CK、TX 和 RX 复用功能在 GPIO 端口的映像。 0: 没有重映像(CTS/PA0，RTS/PA1，TX/PA2，RX/PA3，CK/PA4)； 1: 重映像(CTS/PD3，RTS/PD4，TX/PD5，RX/PD6，CK/PD7)；

位 2	**USART1_REMAP**：USART1 的重映像(USART1 remapping) 该位可由软件置’1’或置’0’，控制 USART1 的 TX 和 RX 复用功能在 GPIO 端口的映像。 0: 没有重映像(TX/PA9，RX/PA10)； 1: 重映像(TX/PB6，RX/PB7)。
位 1	**I2C1_REMAP**：I2C1 的重映像(I2C1 remapping) 该位可由软件置’1’或置’0’，控制 I2C1 的 SCL 和 SDA 复用功能在 GPIO 端口的映像。 0: 没有重映像(SCL/PB6，SDA/PB7)； 1: 重映像(SCL/PB8，SDA/PB9)。
位 0	**SPI1_REMAP**：SPI1 的重映像 该位可由软件置’1’或置’0’，控制 SPI1 的 NSS、SCK、MISO 和 MOSI 复用功能在 GPIO 端口的映像。 0: 没有重映像(NSS/PA4，SCK/PA5，MISO/PA6，MOSI/PA7)； 1: 重映像(NSS/PA15，SCK/PB3，MISO/PB4，MOSI/PB5)。

8.4.3 外部中断配置寄存器 1(AFIO_EXTICR1)

地址偏移：0x08

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16	保留。
位 15:0	**EXTIx[3:0]**：EXTIx 配置(x = 0 … 3) (EXTI x configuration) 这些位可由软件读写，用于选择 EXTIx 外部中断的输入源。参看 10.2.5 节。 0000：PA[x]引脚 0001：PB[x]引脚 0010：PC[x]引脚 0011：PD[x]引脚

8.4.4 外部中断配置寄存器 2(AFIO_EXTICR2)

地址偏移：0x0C

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16	保留。

位 15:0	**EXTIx[3:0]**：EXTIx 配置(x = 4 … 7) (EXTI x configuration) 这些位可由软件读写，用于选择 EXTIx 外部中断的输入源。 0000：PA[x]引脚 0001：PB[x]引脚 0010：PC[x]引脚 0011：PD[x]引脚

8.4.5 外部中断配置寄存器 3(AFIO_EXTICR3)

地址偏移：0x10

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16	保留。
位 15:0	**EXTIx[3:0]**：EXTIx 配置(x = 8 … 11) (EXTI x configuration) 这些位可由软件读写，用于选择 EXTIx 外部中断的输入源。 0000：PA[x]引脚 0001：PB[x]引脚 0010：PC[x]引脚 0011：PD[x]引脚

8.4.6 外部中断配置寄存器 4(AFIO_EXTICR4)

地址偏移：0x14

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16	保留。
位 15:0	**EXTIx[3:0]**：EXTIx 配置(x = 12 … 15) (EXTI x configuration) 这些位可由软件读写，用于选择 EXTIx 外部中断的输入源。 0000：PA[x]引脚 0001：PB[x]引脚 0010：PC[x]引脚 0011：PD[x]引脚

8.5 GPIO 和 AFIO 寄存器地址映象

关于寄存器起始地址，请参考表 1。下面列出了 GPIO 和 AFIO 寄存器映象和复位数值。

表 35 GPIO 寄存器地址映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	GPIOx_CRL	CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]		CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
	复位值	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
004h	GPIOx_CRH	CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]		CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
	复位值											0		1	0	1	0	1	0	1	0	1	0	1		0	1	0	1	0	1		
008h	GPIOx_IDR																									IDR[15:0]						0	1
	复位值																	0		0		0		0		0		0		0			
00Ch	GPIOx_ODR																									ODR[15:0]						0	0
	复位值																	0		0		0		0		0		0		0			
010h	GPIOx_BSRR	BR[15:0]																BSR[15:0]														0	0
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
014h	GPIOx_BRR	保留															LCK	BR[15:0]														0	0
	复位值																	0	0	0	0	0	0	0	0								
018h	GPIOx_LCKR	保留																LCK[15:0]														0	0
	复位值																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

表 36 AFIO 寄存器地址映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	AFIO_EVCR	保留																								EVOE	PORT[2:0]			PIN[3:0]			
	复位值																									0	0	0	0	0	0	0	0
004h	AFIO_MAPR	保留				SWJ_CFG[2:0]			保留				ADC2_ETRGREG_REMA	ADC2_ETRGINJ_REMA	ADC1_ETRGREG_REMA	ADC1_ETRGINJ_REMA	-	PD01_REMAP	CAN_REMAP[1:0]		TIM4_REMAP	TIM3_REMAP[1:0]		TIM2_REMAP[1:0]		TIM1_REMAP[1:0]		USART3_REMAP[1:0]		USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP
	复位值					0	0	0					0	0	0	0		0	0		0	0		0		0		0		0	0	0	0
008h	AFIO_EXTIC	保留																EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	AFIO_EXTICR2																	EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	AFIO_EXTICR3																	EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	AFIO_EXTICR4																	EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

9 中断和事件

9.1 嵌套向量中断控制器

特性

- 44 个可屏蔽中断通道(不包含 16 个 Cortex™-M3 的中断线)；
- 16 个可编程的优先等级(使用了 4 位中断优先级)；
- 低延迟的异常和中断处理；
- 电源管理控制；
- 系统控制寄存器的实现；

嵌套向量中断控制器(NVIC)和处理器核的接口紧密相连，可以实现低延迟的中断处理和高效地处理晚到的中断。

嵌套向量中断控制器管理着包括内核异常等中断。更多关于异常和 NVIC 编程的说明请参考 Cortex-M3 相关手册。

9.1.1 系统嘀嗒(SysTick)校准值寄存器

系统嘀嗒校准值固定为 9000，当系统嘀嗒时钟设定为 9MHz(HCLK/8 的最大值)，产生 1ms 时间基准。

9.1.2 中断和异常向量

下面表格，列出了 HK32F10xxx 产品的向量表。

表 37 HK32F10xxx 产品的向量表

位置	优先级	优先级类型	名称	说明	地址
	-	-	-	保留	0x0000_0000
	-3	固定	Reset	复位	0x0000_0004
	-2	固定	NMI	不可屏蔽中断 RCC 时钟安全系统(CSS)联接到 NMI 向量	0x0000_0008
	-1	固定	硬件失效(HardFault)	所有类型的失效	0x0000_000C
	0	可设置	存储管理(MemManage)	存储器管理	0x0000_0010
	1	可设置	总线错误(BusFault)	预取指失败，存储器访问失败	0x0000_0014
	2	可设置	错误应用(UsageFault)	未定义的指令或非法状态	0x0000_0018
	-	-	-	保留	0x0000_001C ~0x0000_002B
	3	可设置	SVCall	通过 SWI 指令的系统服务调用	0x0000_002C
	4	可设置	调试监控(DebugMonitor)	调试监控器	0x0000_0030
	-	-	-	保留	0x0000_0034
	5	可设置	PendSV	可挂起的系统服务	0x0000_0038
	6	可设置	SysTick	系统嘀嗒定时器	0x0000_003C
0	7	可设置	WWDG	窗口定时器中断	0x0000_0040
1	8	可设置	PVD	连到 EXTI 的电源电压检测(PVD)中断	0x0000_0044
2	9	可设置	TAMPER	侵入检测中断	0x0000_0048

3	10	可设置	RTC	实时时钟(RTC)全局中断	0x0000_004C
4	11	可设置	FLASH	闪存全局中断	0x0000_0050
5	12	可设置	RCC	复位和时钟控制(RCC)中断	0x0000_0054
6	13	可设置	EXTI0	EXTI 线 0 中断	0x0000_0058
7	14	可设置	EXTI1	EXTI 线 1 中断	0x0000_005C
8	15	可设置	EXTI2	EXTI 线 2 中断	0x0000_0060
9	16	可设置	EXTI3	EXTI 线 3 中断	0x0000_0064
10	17	可设置	EXTI4	EXTI 线 4 中断	0x0000_0068
11	18	可设置	DMA1 通道 1	DMA1 通道 1 全局中断	0x0000_006C
12	19	可设置	DMA1 通道 2	DMA1 通道 2 全局中断	0x0000_0070
13	20	可设置	DMA1 通道 3	DMA1 通道 3 全局中断	0x0000_0074
14	21	可设置	DMA1 通道 4	DMA1 通道 4 全局中断	0x0000_0078
15	22	可设置	DMA1 通道 5	DMA1 通道 5 全局中断	0x0000_007C
16	23	可设置	DMA1 通道 6	DMA1 通道 6 全局中断	0x0000_0080
17	24	可设置	DMA1 通道 7	DMA1 通道 7 全局中断	0x0000_0084
18	25	可设置	ADC1_2	ADC1 和 ADC2 的全局中断	0x0000_0088
19	26	可设置	USB_HP_CAN_TX	USB 高优先级或 CAN 发送中断	0x0000_008C
20	27	可设置	USB_LP_CAN_RX0	USB 低优先级或 CAN 接收 0 中断	0x0000_0090
21	28	可设置	CAN_RX1	CAN 接收 1 中断	0x0000_0094
22	29	可设置	CAN_SCE	CAN SCE 中断	0x0000_0098
23	30	可设置	EXTI9_5	EXTI 线[9:5]中断	0x0000_009C
24	31	可设置	TIM1_BRK	TIM1 刹车中断	0x0000_00A0
25	32	可设置	TIM1_UP	TIM1 更新中断	0x0000_00A4
26	33	可设置	TIM1_TRG_COM	TIM1 触发和通信中断	0x0000_00A8
27	34	可设置	TIM1_CC	TIM1 捕获比较中断	0x0000_00AC
28	35	可设置	TIM2	TIM2 全局中断	0x0000_00B0
29	36	可设置	TIM3	TIM3 全局中断	0x0000_00B4
30	37	可设置	TIM4	TIM4 全局中断	0x0000_00B8
31	38	可设置	I2C1_EV	I^2C1 事件中断	0x0000_00BC
32	39	可设置	I2C1_ER	I^2C1 错误中断	0x0000_00C0
33	40	可设置	I2C2_EV	I^2C2 事件中断	0x0000_00C4
34	41	可设置	I2C2_ER	I^2C2 错误中断	0x0000_00C8
35	42	可设置	SPI1	SPI1 全局中断	0x0000_00CC
36	43	可设置	SPI2	SPI2 全局中断	0x0000_00D0
37	44	可设置	USART1	USART1 全局中断	0x0000_00D4
38	45	可设置	USART2	USART2 全局中断	0x0000_00D8
39	46	可设置	USART3	USART3 全局中断	0x0000_00DC
40	47	可设置	EXTI15_10	EXTI 线[15:10]中断	0x0000_00E0
41	48	可设置	RTCAlarm	连到 EXTI 的 RTC 闹钟中断	0x0000_00E4
42	49	可设置	USB 唤醒	连到 EXTI 的从 USB 待机唤醒中断	0x0000_00E8
43	50	保留	-	保留	0x0000_00EC
44	51	保留	-	保留	0x0000_00F0
45	52	保留	-	保留	0x0000_00F4

46	53	保留	-	保留	0x0000_00F8
47	54	可设置	COALU	连接到 COALU 中断	0x0000_00FC
48	55	可设置	-	保留	0x0000_0100
49	56	可设置	-	保留	0x0000_0104

9.2 外部中断/事件控制器(EXTI)

外部中断/事件控制器由 19 个能产生事件/中断请求的边沿检测器。每个输入线可以独立地配置输入类型(脉冲或挂起)和对应的触发事件(上升沿或下降沿或者双边沿都触发)。每个输入线都可以独立地被屏蔽。挂起寄存器保持着状态线的中断请求。

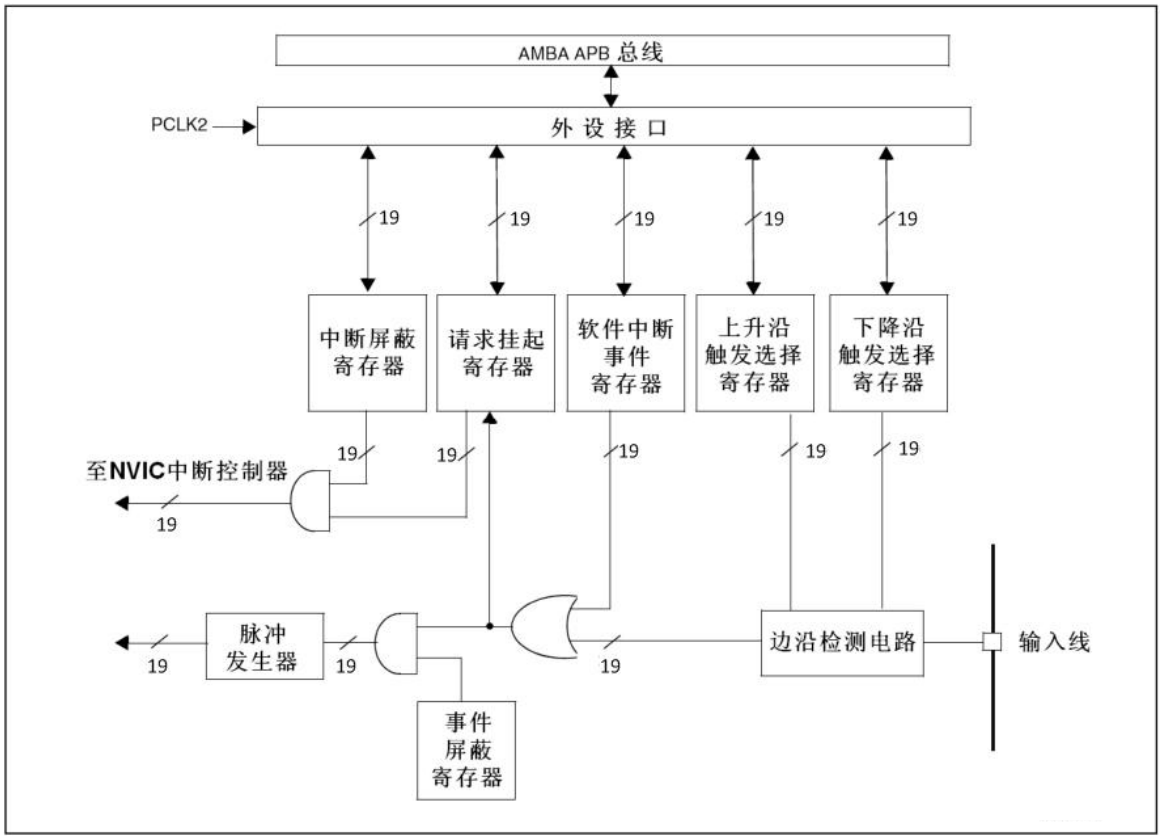
9.2.1 主要特性

EXTI 控制器的主要特性如下：

- 每个中断/事件都有独立的触发和屏蔽
- 每个中断线都有专用的状态位
- 支持多达 19 个软件的中断/事件请求
- 检测脉冲宽度低于 APB2 时钟宽度的外部信号。参见数据手册中电气特性部分的相关参数。

9.2.2 框图

图 15 外部中断/事件控制器框图



9.2.3 唤醒事件管理

HK32F10xxx 可以处理外部或内部事件来唤醒内核(WFE)。唤醒事件可以通过下述配置产生：

- 在外设的控制寄存器使能一个中断，但不在 NVIC 中使能，同时在 Cortex-M3 的系统控制寄存器中使能 SEVONPEND 位。当 CPU 从 WFE 恢复后，需要清除相应外设的中断挂起位和外 设 NVIC 中断通道挂起位(在 NVIC 中断清除挂起寄存器中)。

- 配置一个外部或内部 EXTI 线为事件模式，当 CPU 从 WFE 恢复后，因为对应事件线的挂起位没有被置位，不必清除相应外设的中断挂起位或 NVIC 中断通道挂起位。

使用外部 I/O 端口作为唤醒事件，请参见 10.2.4 节的功能说明

9.2.4 功能说明

要产生中断，必须先配置好并使能中断线。根据需要的边沿检测设置 2 个触发寄存器，同时在中断屏蔽寄存器的相应位写'1'允许中断请求。当外部中断线上发生了期待的边沿时，将产生一个中断请求，对应的挂起位也随之被置'1'。在挂起寄存器的对应位写'1'，将清除该中断请求。

如果需要产生事件，必须先配置好并使能事件线。根据需要的边沿检测通过设置 2 个触发寄存器，同时在事件屏蔽寄存器的相应位写'1'允许事件请求。当事件线上发生了需要的边沿时，将产生一个事件请求脉冲，对应的挂起位不被置'1'。

通过在软件中断/事件寄存器写'1'，也可以通过软件产生中断/事件请求。

硬件中断选择

通过下面的过程来配置 19 个线路做为中断源：

- 配置 19 个中断线的屏蔽位(EXTI_IMR)
- 配置所选中断线的触发选择位(EXTI_RTSR 和 EXTI_FTSR)；
- 配置对应到外部中断控制器(EXTI)的 NVIC 中断通道的使能和屏蔽位，使得 20 个中断线中的请求可以被正确地响应。

硬件事件选择

通过下面的过程，可以配置 19 个线路为事件源

- 配置 19 个事件线的屏蔽位(EXTI_EMR)
- 配置事件线的触发选择位(EXTI_RTSR 和 EXTI_FTSR)

软件中断/事件的选择

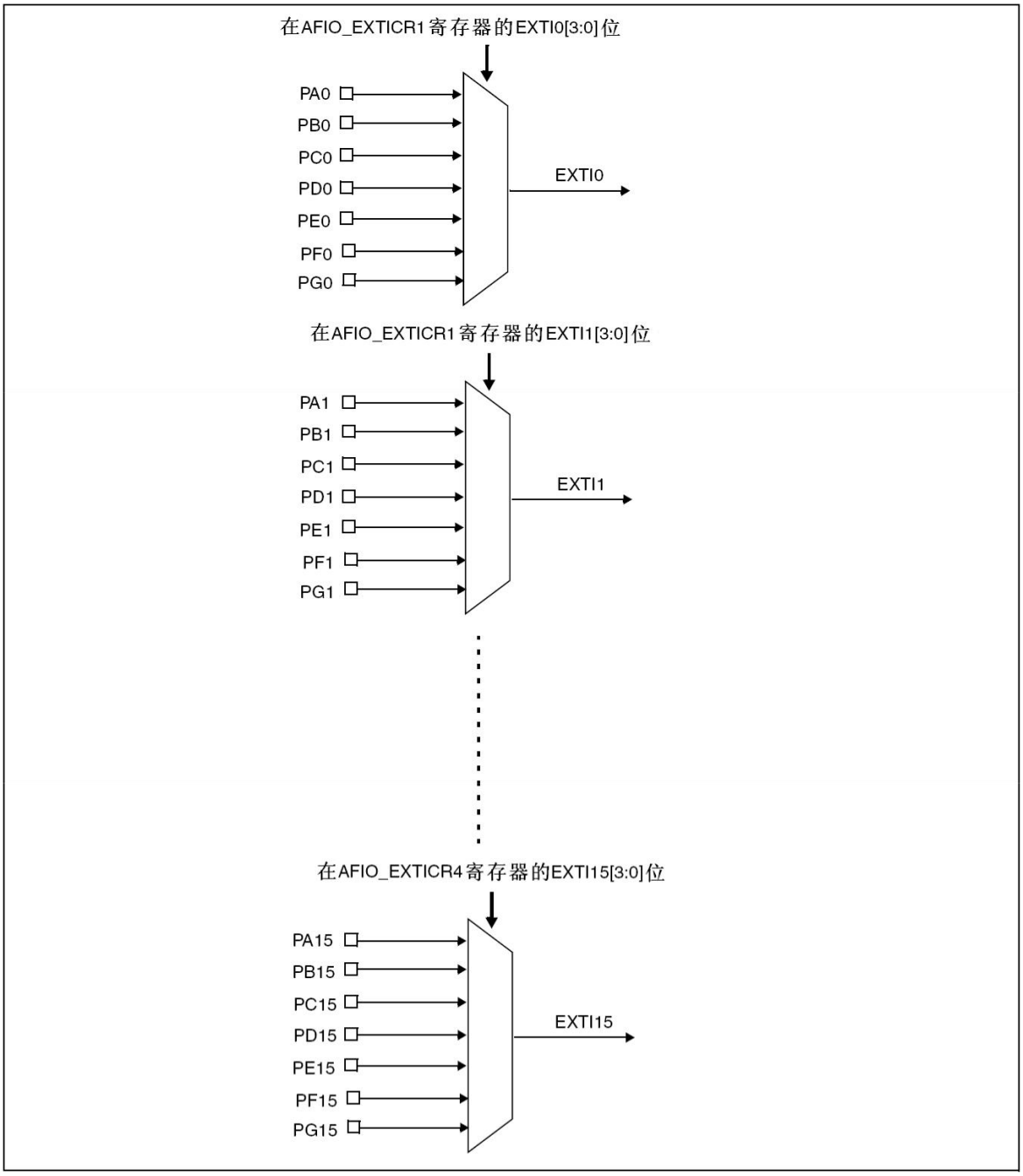
19 个线路可以被配置成软件中断/事件线。下面是产生软件中断的过程：

- 配置 19 个中断/事件线屏蔽位(EXTI_IMR, EXTI_EMR)
- 设置软件中断寄存器的请求位(EXTI_SWIER)

9.2.5 外部中断/事件线路映像

112 通用 I/O 端口以下图的方式连接到 16 个外部中断/事件线上：

图 16 外部中断通用 I/O 映像



1. 通过 AFIO_EXTICRx 配置 GPIO 线上的外部中断/事件，必须先使能 AFIO 时钟。参见 8.3.7 节；。

另外四个 EXTI 线的连接方式如下：

- EXTI 线 16 连接到 PVD 输出
- EXTI 线 17 连接到 RTC 闹钟事件
- EXTI 线 18 连接到 USB 唤醒事件

9.3 EXTI 寄存器描述

关于寄存器描述中的缩略词，请参考 2.1 节。 必须以字(32 位)的方式操作这些外设寄存器。

9.3.1 中断屏蔽寄存器(EXTI_IMR)

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留													MR18	MR17	MR16
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR4	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:19	保留，必须始终保持为复位状态(0)。
位 18:0	**MRx**: 线 x 上的中断屏蔽(Interrupt Mask on line x) 0：屏蔽来自线 x 上的中断请求； 1：开放来自线 x 上的中断请求。

9.3.2 事件屏蔽寄存器(EXTI_EMR)

偏移地址：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留													MR18	MR17	MR16
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR4	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:19	保留，必须始终保持为复位状态(0)。
位 18:0	**MRx**: 线 x 上的事件屏蔽 (Event Mask on line x) 0：屏蔽来自线 x 上的事件请求； 1：开放来自线 x 上的事件请求。

9.3.3 上升沿触发选择寄存器(EXTI_RTSR)

偏移地址：0x08

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留													TR18	TR17	TR16
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:19	保留，必须始终保持为复位状态(0)。
位 18:0	**TRx**: 线 x 上的上升沿触发事件配置位(Rising trigger event configuration bit of line x) 0：禁止输入线 x 上的上升沿触发(中断和事件) 1：允许输入线 x 上的上升沿触发(中断和事件)

注意: 外部唤醒线是边沿触发的，这些线上不能出现毛刺信号。
在写 EXTI_RTSR 寄存器时，在外部中断线上的上升沿信号不能被识别，挂起位也不会被置位。
在同一中断线上，可以同时设置上升沿和下降沿触发。即任一边沿都可触发中断。

9.3.4 下降沿触发选择寄存器(EXTI_FTSR)

偏移地址：0x0C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留													TR18	TR17	TR16
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:19	保留，必须始终保持为复位状态(0)。
位 18:0	**TRx**: 线 x 上的下降沿触发事件配置位(Falling trigger event configuration bit of line x) 0：禁止输入线 x 上的下降沿触发(中断和事件) 1：允许输入线 x 上的下降沿触发(中断和事件)

注意: 外部唤醒线是边沿触发的，这些线上不能出现毛刺信号。在写 EXTI_FTSR 寄存器时，在外部中断线上的下降沿信号不能被识别，挂起位不会被置位。在同一中断线上，可以同时设置上升沿和下降沿触发。即任一边沿都可触发中断。

9.3.5 软件中断事件寄存器(EXTI_SWIER)

偏移地址：0x10

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留													SWIER 18	SWIER 17	SWIER 16
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWIER 15	SWIER 14	SWIER 13	SWIER 12	SWIER 11	SWIER 10	SWIER 9	SWIER 8	SWIER 7	SWIER 6	SWIER 5	SWIER 4	SWIER 3	SWIER 2	SWIER 1	SWIER 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:19	保留，必须始终保持为复位状态(0)。
位 18:0	**SWIERx**: 线 x 上的软件中断(Software interrupt on line x) 当该位为'0'时，写'1'将设置 EXTI_PR 中相应的挂起位。如果在 EXTI_IMR 和 EXTI_EMR 中允许产生该中断，则此时将产生一个中断。 注：通过清除 EXTI_PR 的对应位(写入'1')，可以清除该位为'0'。

9.3.6 挂起寄存器(EXTI_PR)

偏移地址：0x14

复位值：0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留													PR18	PR17	PR16
													rc w1	rc w1	rc w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1

位 31:19	保留，必须始终保持为复位状态(0)。
位 18:0	**PRx**: 挂起位(Pending bit) 0：没有发生触发请求 1：发生了选择的触发请求 当在外部中断线上发生了选择的边沿事件，该位被置'1'。在该位中写入'1'可以清除它，也可以 通过改变边沿检测的极性清除。

9.3.7 外部中断/事件寄存器映像

下表列出了 EXTI 寄存器的映像和复位值。

表 38 外部中断/事件控制器寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	EXTI_IMR	保留											MR[18:0]																				
	复位值												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004h	EXTI_EMR	保留											MR[18:0]																				
	复位值												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
008h	EXTI_RTSR	保留											TR[18:0]																				
	复位值												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	EXTI_FTSR	保留											TR[18:0]																				
	复位值												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	EXTI_SWIER	保留											SWIER[18:0]																				
	复位值												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	EXTI_PR	保留											PR[18:0]																				
	复位值												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

关于寄存器的起始地址，参见表 1。

10 模拟/数字转换（ADC）

10.1 ADC 介绍

12 位 ADC 是一种逐次逼近型模拟数字转换器。它有多达 18 个通道，可测量 16 个外部和 2 个内部信号源。各通道的 A/D 转换可以单次、连续、扫描或间断模式执行。ADC 的结果可以左对齐或右对齐方式存储在 16 位数据寄存器中。

模拟看门狗特性允许应用程序检测输入电压是否超出用户定义的高/低阀值。

ADC 的输入时钟不得超过 14MHz，它是由 PCLK2 经分频产生。见图 7。

10.2 ADC 主要特征

- 12 位分辨率
- 转换结束、注入转换结束和发生模拟看门狗事件时产生中断
- 单次和连续转换模式
- 从通道 0 到通道 n 的自动扫描模式
- 自校准
- 带内嵌数据一致性的数据对齐
- 采样间隔可以按通道分别编程
- 规则转换和注入转换均有外部触发选项
- 间断模式
- 双重模式(带 2 个或以上 ADC 的器件)
- ADC 转换时间：
  - HK32F103xx 增强型产品：时钟为 56MHz 时为 1μs(时钟为 72MHz 为 1.17μs)
- ADC 供电要求：2V 到 3.6V
- ADC 输入范围：$V_{REF-} \leqslant V_{IN} \leqslant V_{REF+}$
- 规则通道转换期间有 DMA 请求产生。

注意：　如果有 V_{REF-} 引脚(取决于封装)，必须和 VSSA 相连接

10.3 ADC 功能描述

下图为一个 ADC 模块的框图，表 39 为 ADC 引脚的说明。

图 17 单个 ADC 框图

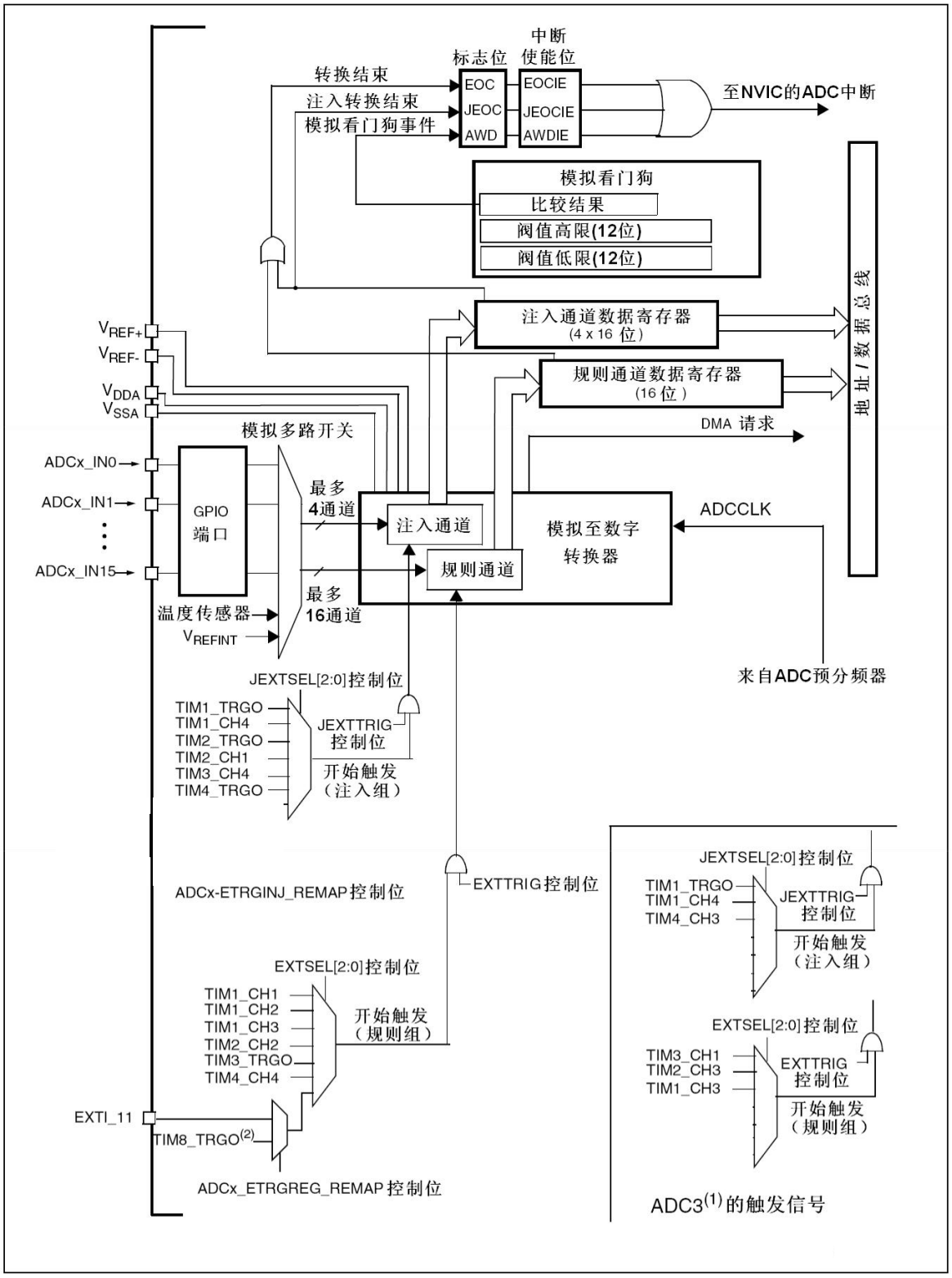


表 39 ADC 引脚

名称	信号类型	注解
V_{REF+}	输入，模拟参考正极	ADC 使用的高端/正极参考电压，2V ≤ V_{REF+} ≤ V_{DDA}
V_{DDA}(1)	输入，模拟电源	等效于 V_{DD} 的模拟电源且：2V ≤ V_{DDA} ≤ V_{DD}(3.6V)
V_{REF-}	输入，模拟参考负极	ADC 使用的低端/负极参考电压，V_{REF-} = V_{SSA}
V_{SSA}(1)	输入，模拟电源地	等效于 V_{SS} 的模拟电源地
ADCx_IN[15:0]	模拟输入信号	16 个模拟输入通道

1. V_{DDA} 和 V_{SSA} 应该分别连接到 V_{DD} 和 V_{SS}。

10.3.1 ADC 开关控制

通过设置 ADC_CR2 寄存器的 ADON 位可给 ADC 上电。当第一次设置 ADON 位时，它将 ADC 从断电状态下唤醒。

ADC 上电延迟一段时间后(t_{STAB})，再次设置 ADON 位时开始进行转换。 通过清除 ADON 位可以停止转换，并将 ADC 置于断电模式。在这个模式中，ADC 几乎不耗电(仅几个μA)。

10.3.2 ADC 时钟

由时钟控制器提供的 ADCCLK 时钟和 PCLK2(APB2 时钟)同步。RCC 控制器为 ADC 时钟提供一个专用的可编程预分频器，详见复位和时钟控制(RCC)章节。

10.3.3 通道选择

有 16 个多路通道。可以把转换组织成两组：规则组和注入组。在任意多个通道上以任意顺序进 行的一系列转换构成成组转换。例如，可以如下顺序完成转换：通道 3、通道 8、通道 2、通道2、通道 0、通道 2、通道 2、通道 15。

- **规则组**由多达 16 个转换组成。规则通道和它们的转换顺序在 ADC_SQRx 寄存器中选择。规则组中转换的总数应写入 ADC_SQR1 寄存器的 L[3:0]位中。
- **注入组**由多达 4 个转换组成。注入通道和它们的转换顺序在 ADC_JSQR 寄存器中选择。注入组里的转换总数目应写入 ADC_JSQR 寄存器的 L[1:0]位中。

如果 ADC_SQRx 或 ADC_JSQR 寄存器在转换期间被更改，当前的转换被清除，一个新的启动脉冲将发送到 ADC 以转换新选择的组。

温度传感器/ VREFINT 内部通道

温度传感器和通道 ADC1_IN16 相连接，内部参照电压 V_{REFINT} 和 ADC1_IN17 相连接。可以按注入或规则通道对这两个内部通道进行转换。

注意： 温度传感器和 V_{REFINT} 只能出现在主 ADC1 中。

10.3.4 单次转换模式

单次转换模式下，ADC 只执行一次转换。该模式既可通过设置 ADC_CR2 寄存器的 ADON 位(只适用于规则通道)启动也可通过外部触发启动(适用于规则通道或注入通道)，这时 CONT 位为 0。

一旦选择通道的转换完成：

- 如果一个规则通道被转换：
  - 转换数据被储存在 16 位 ADC_DR 寄存器中
  - EOC(转换结束)标志被设置
  - 如果设置了 EOCIE，则产生中断。
- 如果一个注入通道被转换：
  - 转换数据被储存在 16 位的 ADC_DRJ1 寄存器中
  - JEOC(注入转换结束)标志被设置

– 如果设置了 JEOCIE 位，则产生中断。

然后 ADC 停止。

10.3.5 连续转换模式

在连续转换模式中，当前面 ADC 转换一结束马上就启动另一次转换。此模式可通过外部触发启动或通过设置 ADC_CR2 寄存器上的 ADON 位启动，此时 CONT 位是 1。

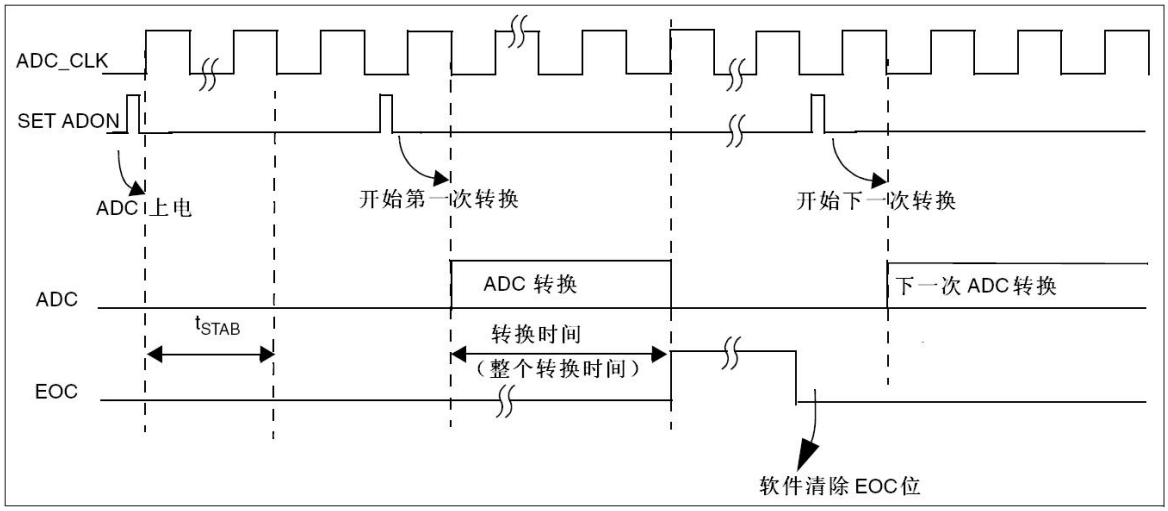
每个转换后：

- 如果一个规则通道被转换：
  - 转换数据被储存在 16 位的 ADC_DR 寄存器中
  - EOC(转换结束)标志被设置
  - 如果设置了 EOCIE，则产生中断。
- 如果一个注入通道被转换：
  - 转换数据被储存在 16 位的 ADC_DRJ1 寄存器中
  - JEOC(注入转换结束)标志被设置
  - 如果设置了 JEOCIE 位，则产生中断。

10.3.6 时序图

如下图所示，ADC 在开始精确转换前需要一个稳定时间 t_{STAB}。在开始 ADC 转换和 14 个时钟周期后，EOC 标志被设置，16 位 ADC 数据寄存器包含转换的结果。

图 18 时序图



10.3.7 模拟看门狗

如果被 ADC 转换的模拟电压低于低阀值或高于高阀值，AWD 模拟看门狗状态位被设置。阀值位 于 ADC_HTR 和 ADC_LTR 寄存器的最低 12 个有效位中。通过设置 ADC_CR1 寄存器的 AWDIE 位 以允许产生相应中断。

阀值独立于由 ADC_CR2 寄存器上的 ALIGN 位选择的数据对齐模式。比较是在对齐之前完成的(见 11.5 节)。 通过配置 ADC_CR1 寄存器，模拟看门狗可以作用于 1 个或多个通道，如表 40 所示。

图 19 模拟看门狗警戒区

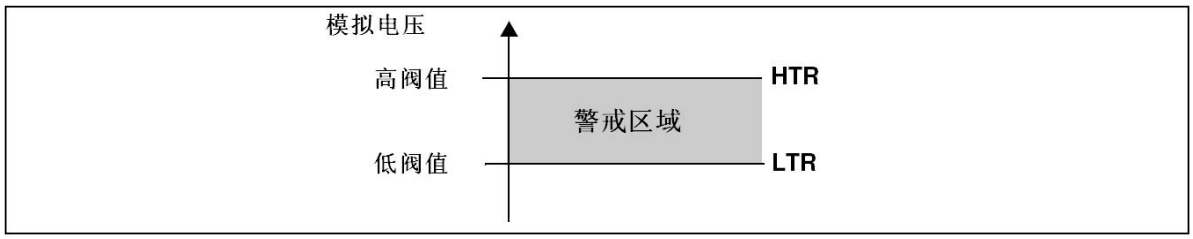


表 40 模拟看门狗通道选择

模拟看门狗警戒的通道	ADC_CR1 寄存器控制位		
	AWDSGL 位	AWDEN 位	JAWDEN 位
无	任意值	0	0
所有注入通道	0	0	1
所有规则通道	0	1	0
所有注入和规则通道	0	1	1
单一的(1)注入通道	1	0	1
单一的(1)规则通道	1	1	0
单一的(1)注入或规则通道	1	1	1

(1) 由 AWDCH[4:0]位选择

10.3.8 扫描模式

此模式用来扫描一组模拟通道。扫描模式可通过设置 ADC_CR1 寄存器的 SCAN 位来选择。一旦这个位被设置，ADC 扫描所有被 ADC_SQRX 寄存器(对规则通道)或 ADC_JSQR(对注入通道)选中的所有通道。在每个组的每个通道上执行单次转换。在每个转换结束时，同一组的下一个通道被自动转换。如果设置了 CONT 位，转换不会在选择组的最后一个通道上停止，而是再次从选择组的第一个通道继续转换。

在扫描模式下，必须设置 ADC_CR2 的 DMA 位，每次规则组的通道转换结束 DR 寄存器被更新的时候，DMA 控制器会把转换结果从 DR 寄存器搬到 SRAM 中。

而注入通道转换的数据总是存储在 ADC_JDRx 寄存器中。

10.3.9 注入通道管理

触发注入

清除 ADC_CR1 寄存器的 JAUTO 位，并且设置 SCAN 位，即可使用触发注入功能。

1. 利用外部触发或通过设置 ADC_CR2 寄存器的 ADON 位，启动一组规则通道的转换。
2. 如果在规则通道转换期间产生一外部注入触发，当前转换被复位，注入通道序列被以单次扫描方式进行转换。
3. 然后，恢复上次被中断的规则组通道转换。如果在注入转换期间产生一规则事件，注入转换不会被中断，但是规则序列将在注入序列结束后被执行。图 20 是其定时图。

注：当使用触发的注入转换时，必须保证触发事件的间隔长于注入序列。例如：序列长度为 28 个 ADC 时钟周期(即 2 个具有 1.5 个时钟间隔采样时间的转换)，触发之间最小的间隔必须是 29 个 ADC 时钟周期。

自动注入

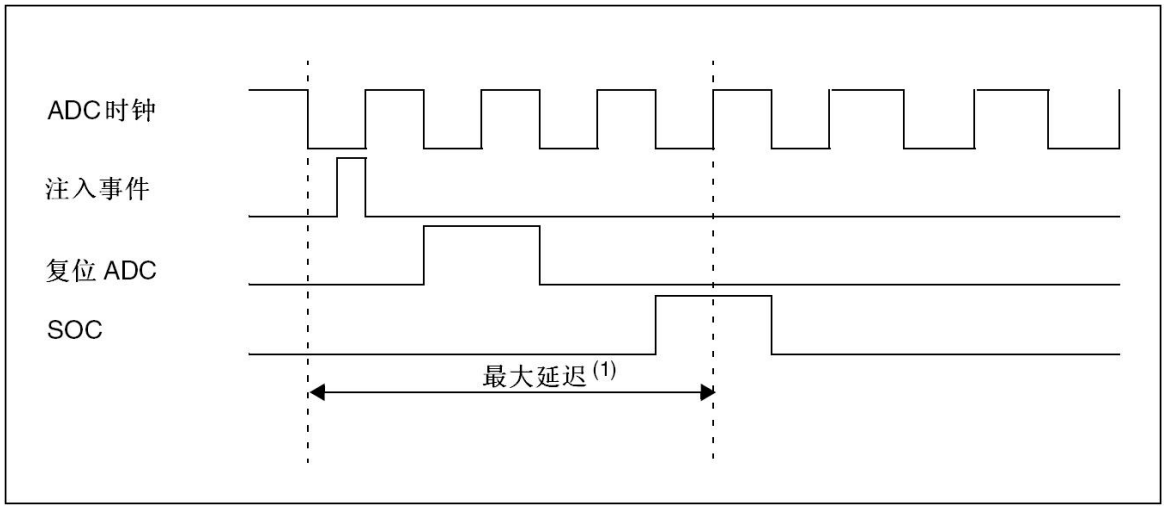
如果设置了 JAUTO 位，在规则组通道之后，注入组通道被自动转换。这可以用来转换在 ADC_SQRx 和 ADC_JSQR 寄存器中设置的多至 20 个转换序列。在此模式里，必须禁

止注入通道的外部触发。如果除 JAUTO 位外还设置了 CONT 位，规则通道至注入通道的转换序列被连续执行。

对于 ADC 时钟预分频系数为 4 至 8 时，当从规则转换切换到注入序列或从注入转换切换到规则序列时，会自动插入 1 个 ADC 时钟间隔；当 ADC 时钟预分频系数为 2 时，则有 2 个 ADC 时钟间隔的延迟。

注意：　不可能同时使用自动注入和间断模式。

图 20 注入转换延时



(1) 最大延迟数值请参考 HK32F103xx 数据手册中有关电气特性部分。

10.3.10 间断模式

规则组

此模式通过设置 ADC_CR1 寄存器上的 DISCEN 位激活。它可以用来执行一个短序列的 n 次转换(n<=8)，此转换是 ADC_SQRx 寄存器所选择的转换序列的一部分。数值 n 由 ADC_CR1 寄存器的DISCNUM[2:0]位给出。

一个外部触发信号可以启动 ADC_SQRx 寄存器中描述的下一轮 n 次转换，直到此序列所有的转换完成为止。总的序列长度由 ADC_SQR1 寄存器的 L[3:0]定义。

举例：

n=3，被转换的通道= 0、1、2、3、6、7、9、10

第一次触发：转换的序列为0、1、2

第二次触发：转换的序列为3、6、7

第三次触发：转换的序列为9、10，并产生 EOC 事件

第四次触发：转换的序列0、1、2

注意：　当以间断模式转换一个规则组时，转换序列结束后不自动从头开始。 当所有子组被转换完成，下一次触发启动第一个子组的转换。在上面的例子中，第四次触发重 新转换第一子组的通道 0、1 和 2。

注入组

此模式通过设置 ADC_CR1 寄存器的 JDISCEN 位激活。在一个外部触发事件后，该模式按通道顺序逐个转换 ADC_JSQR 寄存器中选择的序列。

一个外部触发信号可以启动 ADC_JSQR 寄存器选择的下一个通道序列的转换，直到序列中所有的转换完成为止。总的序列长度由 ADC_JSQR 寄存器的 JL[1:0]位定义。

例子：

n=1，被转换的通道= 1、2、3

第一次触发：通道 1 被转换

第二次触发：通道 2 被转换

第三次触发：通道 3 被转换，并且产生 EOC 和 JEOC 事件

第四次触发：通道 1 被转换

注意：1 当完成所有注入通道转换，下个触发启动第 1 个注入通道的转换。在上述例子中，第四个触发重新转换第 1 个注入通道 1。

2 不能同时使用自动注入和间断模式。

3 必须避免同时为规则和注入组设置间断模式。间断模式只能作用于一组转换。

10.4 校准

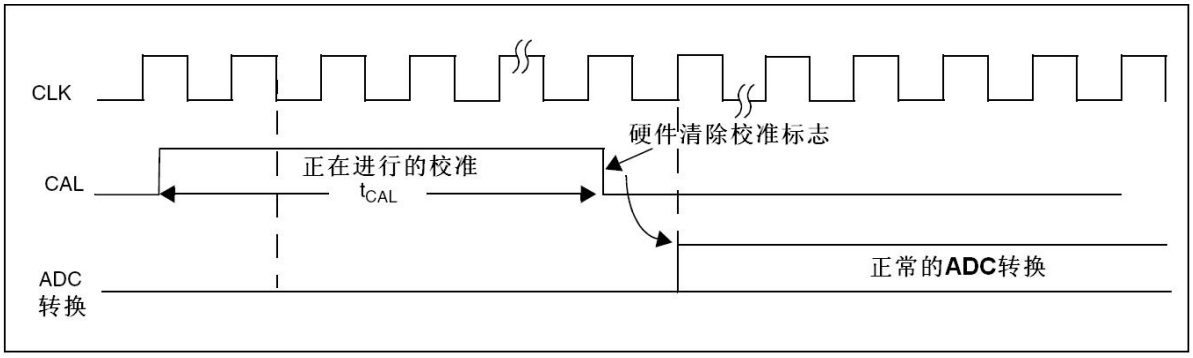
ADC 有一个内置自校准模式。校准可大幅减小因内部电容器组的变化而造成的准精度误差。在校准期间，在每个电容器上都会计算出一个误差修正码(数字值)，这个码用于消除在随后的转换中每个电容器上产生的误差。

通过设置 ADC_CR2 寄存器的 CAL 位启动校准。一旦校准结束，CAL 位被硬件复位，可以开始正常转换。建议在上电时执行一次 ADC 校准。校准阶段结束后，校准码储存在 ADC_DR 中。

注意：1 建议在每次上电后执行一次校准。

2 启动校准前，ADC 必须处于开电状态(ADON='1')超过至少两个 ADC 时钟周期。

图 21 校准时序图



10.5 数据对齐

ADC_CR2 寄存器中的 ALIGN 位选择转换后数据储存的对齐方式。数据可以左对齐或右对齐，如图 22 和图 23 所示。

注入组通道转换的数据值已经减去了在 ADC_JOFRx 寄存器中定义的偏移量，因此结果可以是一个负值。SEXT 位是扩展的符号值。

对于规则组通道，不需减去偏移值，因此只有 12 个位有效。

图 22 数据右对齐

注入组

SEXT	SEXT	SEXT	SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

规则组

0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

图 23 数据左对齐

注入组

SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0

规则组

D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0

10.6 可编程的通道采样时间

ADC 使用若干个 ADC_CLK 周期对输入电压采样，采样周期数目可以通过 ADC_SMPR1 和 ADC_SMPR2 寄存器中的 SMP[2:0]位更改。每个通道可以分别用不同的时间采样。

总转换时间如下计算：

T_{CONV} = 采样时间+ 12.5 个周期

例如：

当 ADCCLK=14MHz，采样时间为 1.5 周期

T_{CONV} = 1.5 + 12.5 = 14 周期 =1μs

10.7 外部触发转换

转换可以由外部事件触发(例如定时器捕获，EXTI 线)。如果设置了 EXTTRIG 控制位，则外部事件就能够触发转换。EXTSEL[2:0]和 JEXTSEL2:0]控制位允许应用程序选择 8 个可能的事件中的 某一个，可以触发规则和注入组的采样。

注意： *当外部触发信号被选为 ADC 规则或注入转换时，只有它的上升沿可以启动转换。*

表 41 ADC1 用于规则通道的外部触发

触发源	类型	EXTSEL[2:0]
TIM1_CC1 事件	来自片上定时器的内部信号	000
TIM1_CC2 事件		001
TIM1_CC3 事件		010
TIM2_CC2 事件		011
TIM3_TRGO 事件		100
TIM4_CC4 事件		101
EXTI 线 11	外部引脚/来自片上定时器的内部信号	110
SWSTART	软件控制位	111

表 42 ADC1 用于注入通道的外部触发

触发源	连接类型	JEXTSEL[2:0]
TIM1_TRGO 事件	来自片上定时器的内部信号	000
TIM1_CC4 事件		001
TIM2_TRGO 事件		010
TIM2_CC1 事件		011
TIM3_CC4 事件		100
TIM4_TRGO 事件		101
EXTI 线 15	外部引脚/来自片上定时器的内部信号	110
JSWSTART	软件控制位	111

表 43 ADC2 用于规则通道的外部触发

触发源	类型	EXTSEL[2:0]
TIM1_CC1 事件	来自片上定时器的内部信号	000
TIM1_CC2 事件		001
TIM1_CC3 事件		010
TIM2_CC2 事件		011
TIM3_TRGO 事件		100
TIM4_CC4 事件		101
EXTI 线 11	外部引脚/来自片上定时器的内部信号	110
SWSTART	软件控制位	111

表 44 ADC2 用于注入通道的外部触发

触发源	连接类型	JEXTSEL[2:0]
TIM1_TRGO 事件	来自片上定时器的内部信号	000
TIM1_CC4 事件		001
TIM2_TRGO 事件		010
TIM2_CC1 事件		011
TIM3_CC4 事件		100
TIM4_TRGO 事件		101
EXTI 线 15	外部引脚/来自片上定时器的内部信号	110
JSWSTART	软件控制位	111

软件触发事件可以通过对寄存器 ADC_CR2 的 SWSTART 或 JSWSTART 位置'1'产生。规则组的转换可以被注入触发打断。

10.8 DMA 请求

因为规则通道转换的值储存在一个仅有的数据寄存器中，所以当转换多个规则通道时需要使用 DMA，这可以避免丢失已经存储在 ADC_DR 寄存器中的数据。

只有在规则通道的转换结束时才产生 DMA 请求，并将转换的数据从 ADC_DR 寄存器传输到用户指定的目的地址。

注： *只有 ADC1 拥有 DMA 功能。由 ADC2 转化的数据可以通过双 ADC 模式，利用 ADC1 的 DMA 功能传输。*

10.9 双 ADC 模式

在有 2 个或以上 ADC 模块的产品中，可以使用双 ADC 模式(见图 24 双 ADC 框图)。 在双 ADC 模式里，根据 ADC1_CR1 寄存器中 DUALMOD[2:0]位所选的模式，转换的启动可以是 ADC1 主和 ADC2 从的交替触发或同步触发。

注意： *在双 ADC 模式里，当转换配置成由外部事件触发时，用户必须将其设置成仅触发主 ADC，从 ADC 设置成软件触发，这样可以防止意外的触发从转换。但是，主和从 ADC 的外部触发必须同时被激活。*

共有 6 种可能的模式：

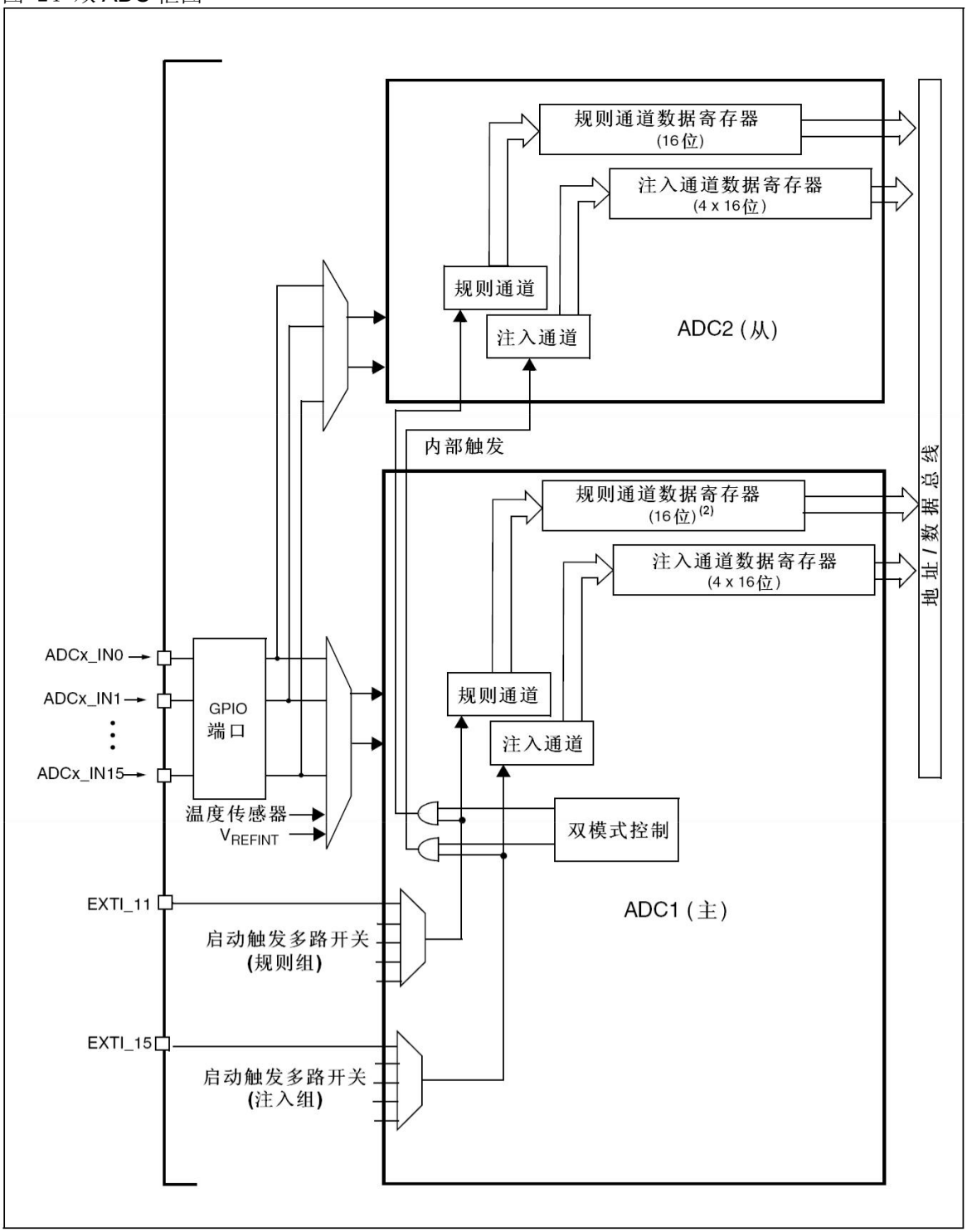
- 同步注入模式
- 同步规则模式
- 快速交叉模式
- 慢速交叉模式

- 交替触发模式
- 独立模式 还有可以用下列方式组合

使用上面的模式：

- 同步注入模式+ 同步规则模式
- 同步规则模式+ 交替触发模式
- 同步注入模式+ 交叉模式

图 24 双 ADC 框图



1．外部触发信号作用于 ADC2，但在本图中没有显示。
2．在某些双 ADC 模式中，在完整的 ADC1 数据寄存器(ADC1_DR)中包含了 ADC1 和 ADC2 的规则转换数据。

10.9.1 同步注入模式

此模式转换一个注入通道组。外部触发来自 ADC1 的注入组多路开关(由 ADC1_CR2 寄存器的 JEXTSEL[2:0]选择)，它同时给 ADC2 提供同步触发。

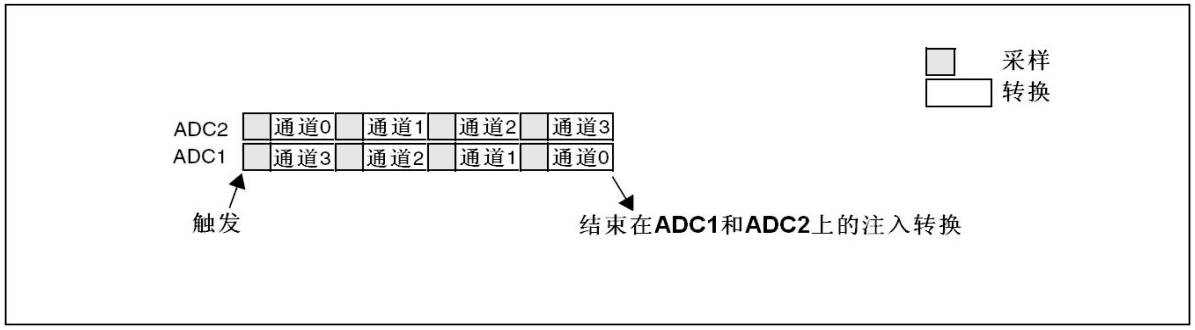
注意：不要在 2 个 ADC 上转换相同的通道(两个 ADC 在同一个通道上的采样时间不能重叠)。

在 ADC1 或 ADC2 的转换结束时：

- 转换的数据存储在每个 ADC 接口的 ADC_JDRx 寄存器中。
- 当所有 ADC1/ADC2 注入通道都被转换时，产生 JEOC 中断(若任一 ADC 接口开放了中断)。

注：在同步注入模式中，在 ADC1 和 ADC2 上同时采样的两个通道必须设置同样的采样时间，来保证两个 ADC 的同步。

图 25 在 4 个通道上的同步注入模式



10.9.2 同步规则模式

此模式在规则通道组上执行。外部触发来自 ADC1 的规则组多路开关(由 ADC1_CR2 寄存器的 EXTSEL[2:0]选择)，它同时给 ADC2 提供同步触发。

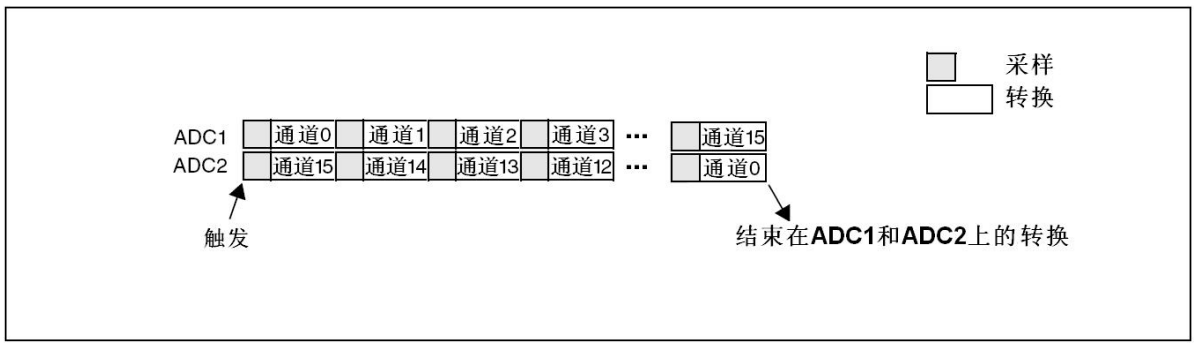
注意：不要在 2 个 ADC 上转换相同的通道((两个 ADC 在同一个通道上的采样时间不能重叠)。

在 ADC1 或 ADC2 的转换结束时：

- 产生一个 32 位 DMA 传输请求(如果设置了 DMA 位)，32 位的 ADC1_DR 寄存器内容传输到 SRAM 中，它上半个字包含 ADC2 的转换数据，低半个字包含 ADC1 的转换数据。
- 当所有 ADC1/ADC2 规则通道都被转换完时，产生 EOC 中断(若任一 ADC 接口开放了中断)。

注：在同步规则模式中，在 ADC1 和 ADC2 上同时采样的两个通道必须设置同样的采样时间，来保证两个 ADC 的同步。

图 26 在 16 个通道上的同步规则模式



10.9.3 快速交叉模式

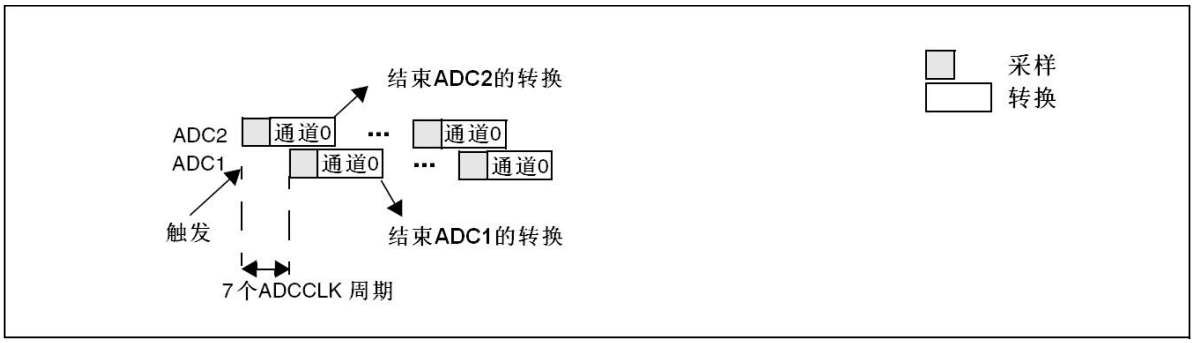
此模式只适用于规则通道组(通常为一个通道)。外部触发来自 ADC1 的规则通道多路开关。外部触发产生后：

- ADC2 立即启动并且
- ADC1 在延迟 7 个 ADC 时钟周期后启动

如果同时设置了 ADC1 和 ADC2 的 CONT 位，所选的两个 ADC 规则通道将被连续地转换。ADC1 产生一个 EOC 中断后(由 EOCIE 使能)，产生一个 32 位的 DMA 传输请求(如果设置了 DMA 位)，ADC1_DR 寄存器的 32 位数据被传输到 SRAM，ADC1_DR 的上半个字包含 ADC2 的转换数据，低半个字包含 ADC1 的转换数据。

注意： *最大允许采样时间<7 个 ADCCLK 周期，避免 ADC1 和 ADC2 转换相同通道时发生两个采样周期的重叠。*

图 27 在 1 个通道上连续转换模式下的快速交叉模式



10.9.4 慢速交叉模式

此模式只适用于规则通道组(只能为一个通道)。外部触发来自 ADC1 的规则通道多路开关。外部触发产生后：

- ADC2 立即启动并且
- ADC1 在延迟 14 个 ADC 时钟周期后启动
- 在延迟第二次 14 个 ADC 周期后 ADC2 再次启动，如此循环。

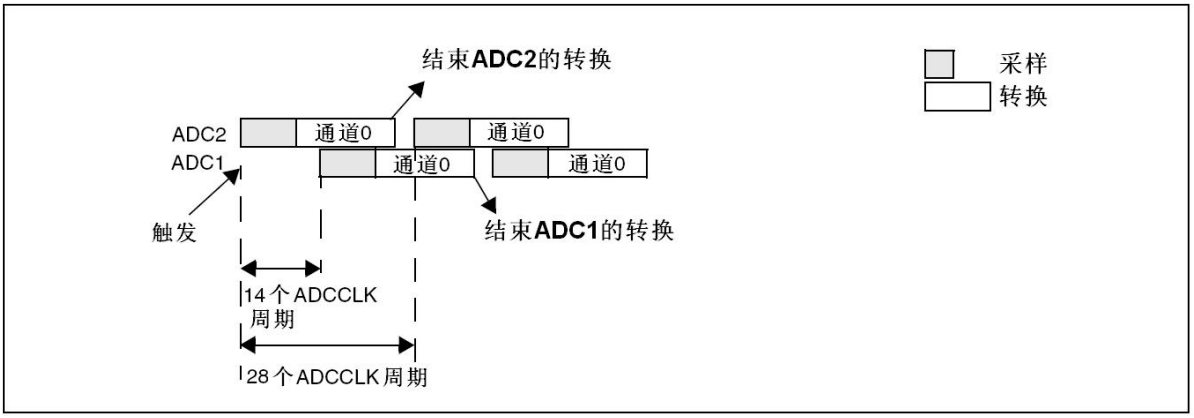
注意： *最大允许采样时间<14 个 ADCCLK 周期，以避免和下个转换重叠。*

ADC1 产生一个 EOC 中断后(由 EOCIE 使能)，产生一个 32 位的 DMA 传输请求(如果设置了 DMA 位)，ADC1_DR 寄存器的 32 位数据被传输到 SRAM，ADC1_DR 的上半个字包含 ADC2 的转换数 据，低半个字包含 ADC1 的转换数据。

在 28 个 ADC 时钟周期后自动启动新的 ADC2 转换。 在这个模式下不能设置 CONT 位，因为它将连续转换所选择的规则通道。

注意： *应用程序必须确保当使用交叉模式时，不能有注入通道的外部触发产生。*

图 28 在 1 个通道上的慢速交叉模式



10.9.5 交替触发模式

此模式只适用于注入通道组。外部触发源来自 ADC1 的注入通道多路开关。

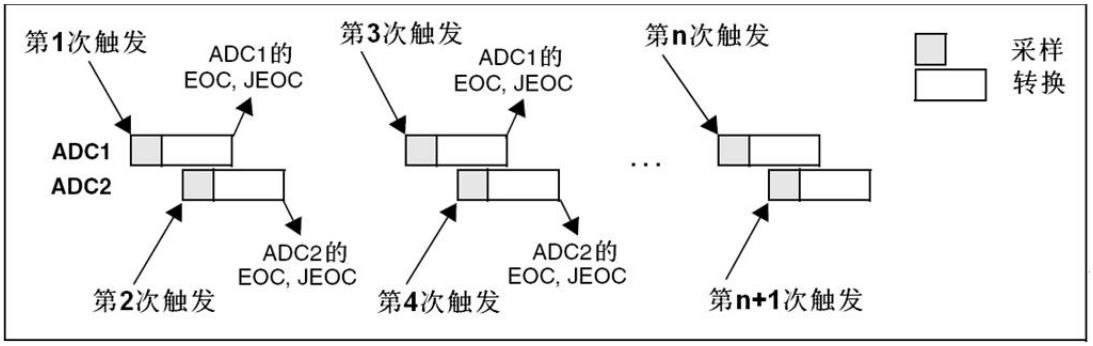
- 当第一个触发产生时，ADC1 上的所有注入组通道被转换。
- 当第二个触发到达时，ADC2 上的所有注入组通道被转换。

- 如此循环……

如果允许产生 JEOC 中断，在所有 ADC1 注入组通道转换后产生一个 JEOC 中断。 如果允许产生 JEOC 中断，在所有 ADC2 注入组通道转换后产生一个 JEOC 中断。

当所有注入组通道都转换完后，如果又有另一个外部触发，交替触发处理从转换 ADC1 注入组通道重新开始。

图 29 交替触发：每个 ADC1 的注入通道组

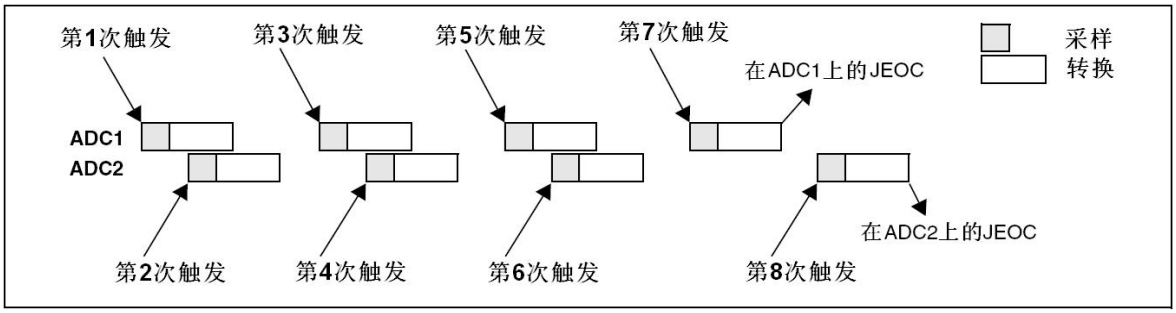


如果 ADC1 和 ADC2 上同时使用了注入间断模式：

- 当第一个触发产生时，ADC1 上的第一个注入通道被转换。
- 当第二个触发到达时，ADC2 上的第一个注入通道被转换。
- 如此循环……

如果允许产生 JEOC 中断，在所有 ADC1 注入组通道转换后产生一个 JEOC 中断。 如果允许产生 JEOC 中断，在所有 ADC2 注入组通道转换后产生一个 JEOC 中断。 当所有注入组通道都转换完后，如果又有另一个外部触发，则重新开始交替触发过程。

图 30 交替触发：在间断模式下每个 ADC 上的 4 个注入通道



10.9.6 独立模式

此模式里，双 ADC 同步不工作，每个 ADC 接口独立工作。

10.9.7 混合的规则/注入同步模式

规则组同步转换可以被中断，以启动注入组的同步转换。

注： *在混合的规则/注入同步模式中，在 ADC1 和 ADC2 上同时采样的两个通道必须设置同样的采样时间，来保证两个 ADC 的同步。*

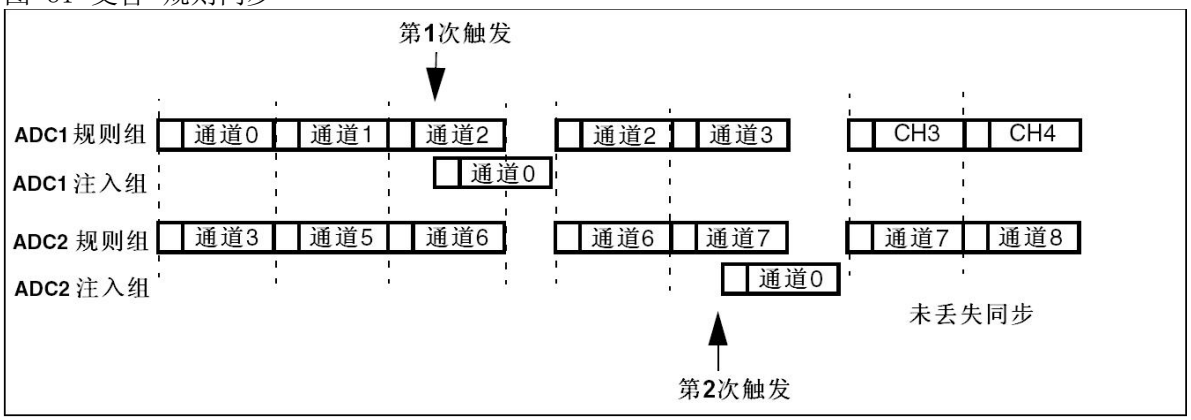
10.9.8 混合的同步规则/交替触发模式

规则组同步转换可以被中断，以启动注入组交替触发转换。图 31 显示了一个规则同步转换被交替触发所中断。

注入交替转换在注入事件到达后立即启动。如果规则转换已经在运行，为了在注入转换后确保同步，所有的 ADC(主和从)的规则转换被停止，并在注入转换结束时同步恢复。

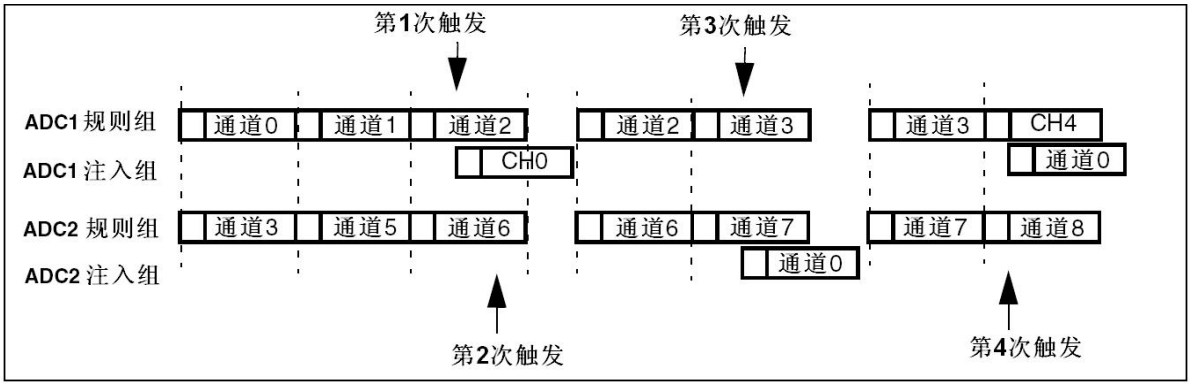
注： *在混合的同步规则/交替触发模式中，在 ADC1 和 ADC2 上同时采样的两个通道必须设置同样的采样时间，来保证两个 ADC 的同步。*

图 31 交替+规则同步



如果触发事件发生在一个中断了规则转换的注入转换期间，这个触发事件将被忽略。下图示出了这种情况的操作(第 2 个触发被忽略)。

图 32 触发事件发生在注入转换期间

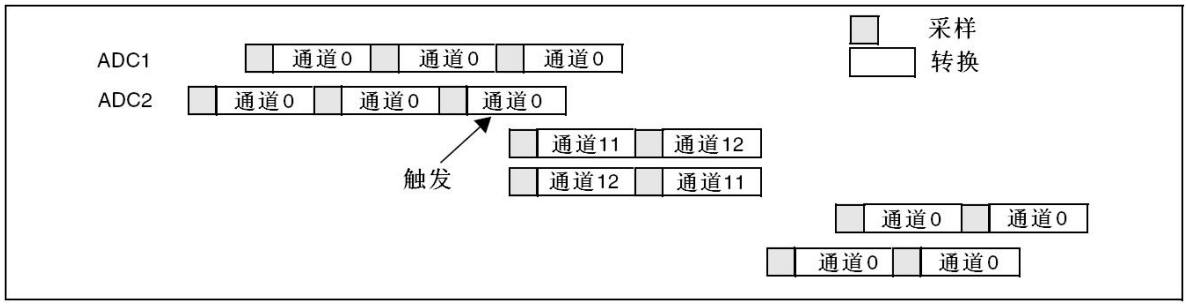


10.9.9 混合同步注入 + 交叉模式

一个注入事件可以中断一个交叉转换。这种情况下，交叉转换被中断，注入转换被启动，在注入序列转换结束时，交叉转换被恢复。下图是这种情况的一个例子。

注：　当 ADC 时钟预分频系数设置为 4 时，交叉模式恢复后不会均匀地分配采样时间，采样间隔是 8 个 ADC 时钟周期与 6 个 ADC 时钟周期轮替，而不是均匀的 7 个 ADC 时钟周期。

图 33 交叉的单通道转换被注入序列 CH11 和 CH12 中断



10.10 温度传感器

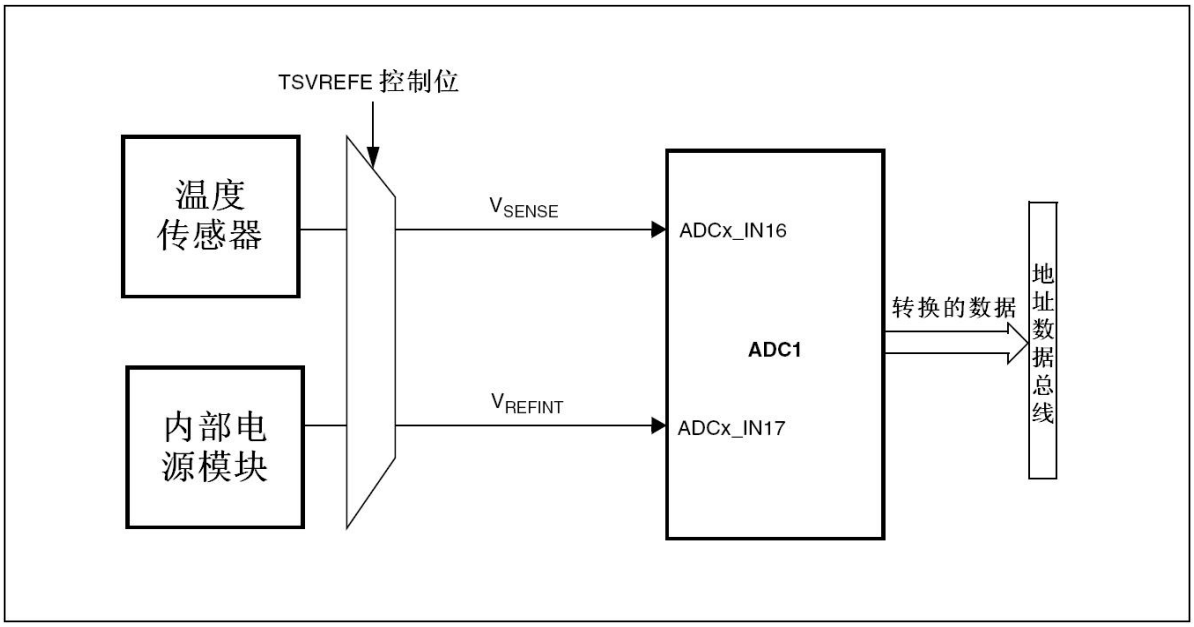
温度传感器可以用来测量器件周围的温度(T_A)。温度传感器在内部和 ADC1_IN16 输入通道相连接，此通道把传感器输出的电压转换成数字值。温度传感器模拟输入推荐采样时间是 17.1μs。

图 34 是温度传感器的方框图。当没有被使用时，传感器可以置于关电模式。

注意：　必须设置 TSVREFE 位激活内部通道：ADC1_IN16(温度传感器)和 ADC1_IN17(VREFINT)的转换。

温度传感器输出电压随温度线性变化，由于生产过程的变化，温度变化曲线的偏移在不同芯片上会有不同(最多相差 45°C)。 内部温度传感器更适合于检测温度的变化，而不是测量绝对的温度。如果需要测量精确的温度，应该使用一个外置的温度传感器。

图 34 温度传感器和 V_{REFINT} 通道框图



读温度

为使用传感器：

1. 选择 ADC1_IN16 输入通道
2. 选择采样时间为 17.1 μs
3. 设置 ADC 控制寄存器 2(ADC_CR2)的 TSVREFE 位，以唤醒关电模式下的温度传感器
4. 通过设置 ADON 位启动 ADC 转换(或用外部触发)
5. 读 ADC 数据寄存器上的 V_{SENSE} 数据结果
6. 利用下列公式得出温度

温度(°C) = {(V_{25} - V_{SENSE}) / Avg_Slope} + 25

这里：

V_{25} = V_{SENSE} 在 25°C 时的数值

Avg_Slope ＝温度与 V_{SENSE} 曲线的平均斜率(单位为 mV/ °C 或μV/ °C)

参考数据手册的电气特性章节中 V_{25} 和 Avg_Slope 的实际值。

注意： 传感器从关电模式唤醒后到可以输出正确水平的 VSENSE 前，有一个建立时间。ADC 在上电后也有一个建立时间，因此为了缩短延时，应该同时设置 ADON 和 TSVREFE 位。

10.11 ADC 中断

规则和注入组转换结束时能产生中断，当模拟看门狗状态位被设置时也能产生中断。它们都有独立的中断使能位。

注： ADC1 和 ADC2 的中断映射在同一个中断向量上。

ADC_SR 寄存器中有 2 个其他标志，但是它们没有相关联的中断：

- JSTRT(注入组通道转换的启动)
- STRT(规则组通道转换的启动)

表 45 ADC 中断

中断事件	事件标志	使能控制位
规则组转换结束	EOC	EOCIE
注入组转换结束	JEOC	JEOCIE
设置了模拟看门狗状态位	AWD	AWDIE

10.12 ADC 寄存器

寄存器描述中使用的一些缩略语请参考 2.1 节。

必须以字(32 位)的方式操作这些外设寄存器。

10.12.1 ADC 状态寄存器(ADC_SR)

地址偏移：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留											STRT	JSTRT	JEOC	EOC	AWD
											rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

位 31:15	保留。必须保持为 0。
位 4	**STRT**：规则通道开始位(Regular channel Start flag) 该位由硬件在规则通道转换开始时设置，由软件清除。 0：规则通道转换未开始； 1：规则通道转换已开始。
位 3	**JSTRT**：注入通道开始位(Injected channel Start flag) 该位由硬件在注入通道组转换开始时设置，由软件清除。 0：注入通道组转换未开始； 1：注入通道组转换已开始。
位 2	**JEOC**：注入通道转换结束位(Injected channel end of conversion) 该位由硬件在所有注入通道组转换结束时设置，由软件清除 0：转换未完成； 1：转换完成。
位 1	**EOC**：转换结束位(End of conversion) 该位由硬件在(规则或注入)通道组转换结束时设置，由软件清除或由读取 ADC_DR 时清除 0：转换未完成； 1：转换完成。
位 0	**AWD**：模拟看门狗标志位(Analog watchdog flag) 该位由硬件在转换的电压值超出了 ADC_LTR 和 ADC_HTR 寄存器定义的范围时设置，由软件清除 0：没有发生模拟看门狗事件； 1：发生模拟看门狗事件。

10.12.2 ADC 控制寄存器 1(ADC_CR1)

地址偏移：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留								AWDEN	JAWDEN	保留		DUALMOD[3:0]			
								rw	rw			rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24	保留。必须保持为 0。
位 23	**AWDEN**：在规则通道上开启模拟看门狗(Analog watchdog enable on regular channels) 该位由软件设置和清除。 0：在规则通道上禁用模拟看门狗； 1：在规则通道上使用模拟看门狗。
位 22	**JAWDEN**：在注入通道上开启模拟看门狗(Analog watchdog enable on injected channels) 该位由软件设置和清除。 0：在注入通道上禁用模拟看门狗； 1：在注入通道上使用模拟看门狗。
位 21:20	保留。必须保持为 0。
位 19:16	**DUALMOD[3:0]**：双模式选择(Dual mode selection) 软件使用这些位选择操作模式。 0000：独立模式 0001：混合的同步规则+注入同步模式 0010：混合的同步规则+交替触发模式 0011：混合同步注入+快速交叉模式 0100：混合同步注入+慢速交叉模式 0101：注入同步模式 0110：规则同步模式 0111：快速交叉模式 1000：慢速交叉模式 1001：交替触发模式 注：在 ADC2 中这些位为保留位 在双模式中，改变通道的配置会产生一个重新开始的条件，这将导致同步丢失。建议在进行任何配置改变前关闭双模式。
位 15:13	**DISCNUM[2:0]**：间断模式通道计数(Discontinuous mode channel count) 软件通过这些位定义在间断模式下，收到外部触发后转换规则通道的数目 000：1 个通道 001：2 个通道 …… 111：8 个通道
位 12	**JDISCEN**：在注入通道上的间断模式(Discontinuous mode on injected channels) 该位由软件设置和清除，用于开启或关闭注入通道组上的间断模式 0：注入通道组上禁用间断模式； 1：注入通道组上使用间断模式。

位 11	**DISCEN**：在规则通道上的间断模式(Discontinuous mode on regular channels) 该位由软件设置和清除，用于开启或关闭规则通道组上的间断模式 0：规则通道组上禁用间断模式； 1：规则通道组上使用间断模式。
位 10	**JAUTO**：自动的注入通道组转换(Automatic Injected Group conversion) 该位由软件设置和清除，用于开启或关闭规则通道组转换结束后自动的注入通道组转换 0：关闭自动的注入通道组转换； 1：开启自动的注入通道组转换。
位 9	**AWDSGL**：扫描模式中在一个单一的通道上使用看门狗 (Enable the watchdog on a channel in scan mode) 该位由软件设置和清除，用于开启或关闭由 AWDCH[4:0]位指定的通道上的模拟看门狗功能 0：在所有的通道上使用模拟看门狗； 1：在单一通道上使用模拟看门狗。
位 8	**SCAN**：扫描模式(Scan mode) 该位由软件设置和清除，用于开启或关闭扫描模式。在扫描模式中，转换由 ADC_SQRx ADC_JSQRx 寄存器选中的通道。 0：关闭扫描模式； 1：使用扫描模式。 注：如果分别设置了 EOCIE 或 JEOCIE 位，只在最后一个通道转换完毕后才会产生 EOC JEOC 中断。
位 7	**JEOCIE**：允许产生注入通道转换结束中断(Interrupt enable for injected channels) 该位由软件设置和清除，用于禁止或允许所有注入通道转换结束后产生中断。 0：禁止 JEOC 中断； 1：允许 JEOC 中断。当硬件设置 JEOC 位时产生中断。
位 6	**AWDIE**：允许产生模拟看门狗中断(Analog watchdog interrupt enable) 该位由软件设置和清除，用于禁止或允许模拟看门狗产生中断。 0：禁止模拟看门狗中断； 1：允许模拟看门狗中断。
位 5	**EOCIE**：允许产生 EOC 中断(Interrupt enable for EOC) 该位由软件设置和清除，用于禁止或允许转换结束后产生中断。 0：禁止 EOC 中断； 1：允许 EOC 中断。当硬件设置 EOC 位时产生中断。
位 4:0	**AWDCH[4:0]**：模拟看门狗通道选择位(Analog watchdog channel select bits) 这些位由软件设置和清除，用于选择模拟看门狗保护的输入通道。 00000：ADC 模拟输入通道 0 00001：ADC 模拟输入通道 1 …… 01111：ADC 模拟输入通道 15 10000：ADC 模拟输入通道 16 10001：ADC 模拟输入通道 17 保留所有其他数值。 注：ADC1 的模拟输入通道 16 和通道 17 在芯片内部分别连到了温度传感器和 V_{REFINT}。 ADC2 的模拟输入通道 16 和通道 17 在芯片内部连到了 VSS。

10.12.3 ADC 控制寄存器 2(ADC_CR2)

地址偏移：0x08

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留								TS VREFE	SW START	JSW START	EXT TRIG	EXTSEL[2:0]			保留
								rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JEXT TRIG	JEXTSEL[2:0]			ALIGN	保留		DMA	保留				RST CAL	CAL	CONT	ADON
rw	rw	rw	rw	rw			rw					rw	rw	rw	rw

位 31:24	保留。必须保持为 0。
位 23	**TSVREFE**：温度传感器和 V_{REFINT} 使能(Temperature sensor and V_{REFINT} enable) 该位由软件设置和清除，用于开启或禁止温度传感器和 V_{REFINT} 通道。在多于 1 个 ADC 的器件中，该位仅出现在 ADC1 中。 0：禁止温度传感器和 V_{REFINT}； 1：启用温度传感器和 V_{REFINT}。
位 22	**SWSTART**：开始转换规则通道(Start conversion of regular channels) 由软件设置该位以启动转换，转换开始后硬件马上清除此位。如果在 EXTSEL[2:0]位中选择了 SWSTART 为触发事件，该位用于启动一组规则通道的转换， 0：复位状态； 1：开始转换规则通道。
位 21	**JSWSTART**：开始转换注入通道(Start conversion of injected channels) 由软件设置该位以启动转换，软件可清除此位或在转换开始后硬件马上清除此位。如 JEXTSEL[2:0]位中选择了 JSWSTART 为触发事件，该位用于启动一组注入通道的转换， 0：复位状态； 1：开始转换注入通道。
位 20	**EXTTRIG**：规则通道的外部触发转换模式 (External trigger conversion mode for channels) 该位由软件设置和清除，用于开启或禁止可以启动规则通道组转换的外部触发事件。 0：不用外部事件启动转换； 1：使用外部事件启动转换。
位 19:17	**EXTSEL[2:0]**：选择启动规则通道组转换的外部事件(External event select for regular group) 这些位选择用于启动规则通道组转换的外部事件 ADC1 和 ADC2 的触发配置如下 000：定时器 1 的 CC1 事件　　100：定时器 3 的 TRGO 事件 001：定时器 1 的 CC2 事件　　101：定时器 4 的 CC4 事件 010：定时器 1 的 CC3 事件　　110：EXTI 线 11 011：定时器 2 的 CC2 事件　　111：SWSTART
位 16	保留。必须保持为 0。
位 15	**JEXTTRIG**：注入通道的外部触发转换模式 (External trigger conversion mode for injected channels) 该位由软件设置和清除，用于开启或禁止可以启动注入通道组转换的外部触发事件。 0：不用外部事件启动转换； 1：使用外部事件启动转换。

位 14:12	**JEXTSEL[2:0]**：选择启动注入通道组转换的外部事件 (External event select for injected group) 这些位选择用于启动注入通道组转换的外部事件。 ADC1 和 ADC2 的触发配置如下 000：定时器 1 的 TRGO 事件　　100：定时器 3 的 CC4 事件 001：定时器 1 的 CC4 事件　　101：定时器 4 的 TRGO 事件 010：定时器 2 的 TRGO 事件　　110：EXTI 线 15 011：定时器 2 的 CC1 事件　　111：JSWSTART
位 11	**ALIGN**：数据对齐(Data alignment) 该位由软件设置和清除。参考图 22 和图 23。 0：右对齐； 1：左对齐。
位 10:9	保留。必须保持为 0。
位 8	**DMA**：直接存储器访问模式(Direct memory access mode) 该位由软件设置和清除。详见 DMA 控制器章节。 0：不使用 DMA 模式； 1：使用 DMA 模式。 注：只有 ADC1 能产生 DMA 请求。
位 7:4	保留。必须保持为 0。
位 3	**RSTCAL**：复位校准(Reset calibration) 该位由软件设置并由硬件清除。在校准寄存器被初始化后该位将被清除。 0：校准寄存器已初始化； 1：初始化校准寄存器。 注：如果正在进行转换时设置 RSTCAL，清除校准寄存器需要额外的周期。
位 2	**CAL**：A/D 校准(A/D Calibration) 该位由软件设置以开始校准，并在校准结束时由硬件清除。 0：校准完成； 1：开始校准。
位 1	**CONT**：连续转换(Continuous conversion) 该位由软件设置和清除。如果设置了此位，则转换将连续进行直到该位被清除。 0：单次转换模式； 1：连续转换模式。
位 0	**ADON**：开/关 A/D 转换器(A/D converter ON / OFF) 该位由软件设置和清除。当该位为'0'时，写入'1'将把 ADC 从断电模式下唤醒。 当该位为'1'时，写入'1'将启动转换。应用程序需注意，在转换器上电至转换开始有一个延迟 t_{STAB}，参见图 18。 0：关闭 ADC 转换/校准，并进入断电模式； 1：开启 ADC 并启动转换。 注：如果在这个寄存器中与 ADON 一起还有其他位被改变，则转换不被触发。 这是为了防止触 发错误的转换。

10.12.4 ADC 采样时间寄存器 1(ADC_SMPR1)

地址偏移：0x0C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留								SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15 0	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24	保留。必须保持为 0。
位 23:0	**SMPx[2:0]**：选择通道 x 的采样时间(Channel x Sample time selection) 这些位用于独立地选择每个通道的采样时间。在采样周期中通道选择位必须保持不变。 000：1.5 周期　　100：41.5 周期 001：7.5 周期　　101：55.5 周期 010：13.5 周期　　110：71.5 周期 011：28.5 周期　　111：239.5 周期 注：ADC1 的模拟输入通道 16 和通道 17 在芯片内部分别连到了温度传感器和 V_{REFINT}。 ADC2 的模拟输入通道 16 和通道 17 在芯片内部连到了 Vss。

10.12.5 ADC 采样时间寄存器 2(ADC_SMPR2)

地址偏移：0x10

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留		SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP5 0	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30	保留。必须保持为 0。
位 29:0	**SMPx[2:0]**：选择通道 x 的采样时间(Channel x Sample time selection) 这些位用于独立地选择每个通道的采样时间。在采样周期中通道选择位必须保持不变。 000：1.5 周期　　100：41.5 周期 001：7.5 周期　　101：55.5 周期 010：13.5 周期　　110：71.5 周期 011：28.5 周期　　111：239.5 周期

10.12.6 ADC 注入通道数据偏移寄存器 x (ADC_JOFRx)(x=1..4)

地址偏移：0x14-0x20

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				JOFFSETx[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:12	保留。必须保持为 0。
位 11:0	**JOFFSETx[11:0]**：注入通道 x 的数据偏移(Data offset for injected channel x) 当转换注入通道时，这些位定义了用于从原始转换数据中减去的数值。转换的结果可以在 ADC_JDRx 寄存器中读出。

10.12.7 ADC 看门狗高阀值寄存器(ADC_HTR)

地址偏移：0x24

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				HT[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:12	保留。必须保持为 0。
位 11:0	**HT[11:0]**：模拟看门狗高阀值(Analog watchdog high threshold) 这些位定义了模拟看门狗的阀值高限。

注：　软件可以在 ADC 转换进行的时候写该寄存器。写入的值在下一次转换结束的时候生效。写入该寄存器存在一定的延迟，导致其生效的准确时间存在一定的不确定性。

10.12.8 ADC 看门狗低阀值寄存器(ADC_LTR)

地址偏移：0x28

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				LT[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:12	保留。必须保持为 0。
位 11:0	**LT[11:0]**：模拟看门狗低阀值(Analog watchdog low threshold) 这些位定义了模拟看门狗的阀值低限。

注：　软件可以在 ADC 转换进行的时候写该寄存器。写入的值在下一次转换结束的时候生效。写入该寄存器存在一定的延迟，导致其生效的准确时间存在一定的不确定性。

10.12.9 ADC 规则序列寄存器

地址偏移：0x2C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留								L[3:0]				SQ16[4:1]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ16_0	SQ15[4:0]					SQ14[4:0]					SQ13[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24	保留。必须保持为 0。
位 23:20	**L[3:0]**：规则通道序列长度(Regular channel sequence length) 这些位由软件定义在规则通道转换序列中的通道数目。 0000： 1 个转换 0001：2 个转换 …… 1111：16 个转换
位 19:15	**SQ16[4:0]**：规则序列中的第 16 个转换(16th conversion in regular sequence) 这些位由软件定义转换序列中的第 16 个转换通道的编号(0~17)。
位 14:10	**SQ15[4:0]**：规则序列中的第 15 个转换(15th conversion in regular sequence)
位 9:5	**SQ14[4:0]**：规则序列中的第 14 个转换(14th conversion in regular sequence)
位 4:0	**SQ13[4:0]**：规则序列中的第 13 个转换(13th conversion in regular sequence)

10.12.10 ADC 规则序列寄存器 2(ADC_SQR2)

地址偏移：0x30

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留		SQ12[4:0]					SQ11[4:0]					SQ10[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ10_0	SQ9[4:0]					SQ8[4:0]					SQ7[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30	保留。必须保持为 0。
位 29:25	**SQ12[4:0]**：规则序列中的第 12 个转换(12th conversion in regular sequence) 这些位由软件定义转换序列中的第 12 个转换通道的编号(0~17)。
位 24:20	**SQ11[4:0]**：规则序列中的第 11 个转换(11th conversion in regular sequence)
位 19:15	**SQ10[4:0]**：规则序列中的第 10 个转换(10th conversion in regular sequence)
位 14:10	**SQ9[4:0]**：规则序列中的第 9 个转换(9th conversion in regular sequence)
位 9:5	**SQ8[4:0]**：规则序列中的第 8 个转换(82th conversion in regular sequence)
位 4:0	**SQ7[4:0]**：规则序列中的第 7 个转换(7th conversion in regular sequence)

10.12.11 ADC 规则序列寄存器 3(ADC_SQR3)

地址偏移：0x34

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留		SQ6[4:0]					SQ5[4:0]					SQ4[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4_0		SQ3[4:0]					SQ2[4:0]					SQ1[4:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30	保留。必须保持为 0。
位 29:25	**SQ6[4:0]**：规则序列中的第 6 个转换(6th conversion in regular sequence) 这些位由软件定义转换序列中的第 6 个转换通道的编号(0~17)。
位 24:20	**SQ5[4:0]**：规则序列中的第 5 个转换(5th conversion in regular sequence)
位 19:15	**SQ4[4:0]**：规则序列中的第 4 个转换(4th conversion in regular sequence)
位 14:10	**SQ3[4:0]**：规则序列中的第 3 个转换(3rd conversion in regular sequence)
位 9:5	**SQ2[4:0]**：规则序列中的第 2 个转换(2nd conversion in regular sequence)
位 4:0	**SQ1[4:0]**：规则序列中的第 1 个转换(1st conversion in regular sequence)

10.12.12 ADC 注入序列寄存器(ADC_JSQR)

地址偏移：0x38

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留										JL[1:0]		JSQ4[4:1]			
										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ4_		JSQ3[4:0]					JSQ2[4:0]					JSQ1[4:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:22	保留。必须保持为 0。
位 21:20	**JL[1:0]**：注入通道序列长度(Injected sequence length) 这些位由软件定义在规则通道转换序列中的通道数目。 00：1 个转换 01：2 个转换 10：3 个转换 11：4 个转换
位 19:15	**JSQ4[4:0]**：注入序列中的第 4 个转换(4th conversion in injected sequence) 这些位由软件定义转换序列中的第 4 个转换通道的编号(0~17)。 注：不同于规则转换序列，如果 JL[1:0]的长度小于 4，则转换的序列顺序是从(4-JL)开始。例如：ADC_JSQR[21:0] = 10 00011 00011 00111 00010，意味着扫描转换将按下列通道顺序转换：7、3、3，而不是 2、7、3。
位 14:10	**JSQ3[4:0]**：注入序列中的第 3 个转换(3rd conversion in injected sequence)
位 9:5	**JSQ2[4:0]**：注入序列中的第 2 个转换(2nd conversion in injected sequence)
位 4:0	**JSQ1[4:0]**：注入序列中的第 1 个转换(1st conversion in injected sequence)

10.12.13 ADC 注入数据寄存器 x (ADC_JDRx) (x= 1..4)

地址偏移：0x3C – 0x48

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:16	保留。必须保持为 0。
位 21:20	**JDATA[15:0]**：注入转换的数据(Injected data) 这些位为只读，包含了注入通道的转换结果。数据是左对齐或右对齐，如图 22 和图 23 所示。

10.12.14 ADC 规则数据寄存器(ADC_DR)

地址偏移：0x4C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC2DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:16	**ADC2DATA[15:0]**：ADC2 转换的数据(ADC2 data) - 在 ADC1 中：双模式下，这些位包含了 ADC2 转换的规则通道数据。见 11.9：双 ADC 模式 - 在 ADC2 中：不使用这些位。
位 15:0	**DATA[15:0]**：规则转换的数据(Regular data) 这些位为只读，包含了规则通道的转换结果。数据是左对齐或右对齐，如图 22 和图 23 所示。

10.12.15 ADC 寄存器地址映像

下表列出了所有的 ADC 寄存器。

表 46 ADC 寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00h	ADC_SR	保留																											STRT	JSTRT	JEOC	EOC	AWD
	复位值																												0	0	0	0	0
04h	ADC_CR1	保留								TSVREFE	AWDEN	JAWDEN	保留	DUALMOD [3:0]				DISC NUM [2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				
	复位值									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08h	ADC_CR2	保留								TSVREFE	SWSTART	JSWSTART	EXTTRIG	EXTSEL [2:0]			保留	JEXTTRIG	JEXTSEL [2:0]			ALIGN	保留		DMA	保留			RSTCAL	CAL	CONT	ADON
	复位值									0	0	0	0	0	0	0		0	0	0	0	0			0				0	0	0	0
0Ch	ADC_SMPR1	保留								采样时间位 SMPx_x																							
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10h	ADC_SMPR2	采样时间位 SMPx_x																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14h	ADC_JOFR1	保留																				0	0	0	JOFFSET1[11:0]								
	复位值																					0	0	0	0	0	0	0	0	0	0	0	0
18h	ADC_JOFR2	保留																				0	0	0	JOFFSET2[11:0]								
	复位值																					0	0	0	0	0	0	0	0	0	0	0	0
1Ch	ADC_JOFR3	保留																				0	0	0	JOFFSET3[11:0]								
	复位值																					0	0	0	0	0	0	0	0	0	0	0	0
20h	ADC_JOFR4	保留																				0	0	0	JOFFSET4[11:0]								
	复位值																					0	0	0	0	0	0	0	0	0	0	0	0
1Ch	ADC_HTR	保留																				0	0	0	HT[11:0]								
	复位值																					0	0	0	0	0	0	0	0	0	0	0	0
20h	ADC_LTR	保留																				0	0	0	LT[11:0]								
	复位值																					0	0	0	0	0	0	0	0	0	0	0	0
2Ch	ADC_SQR1	保留								L[3:0]				规则通道序列 SQx_x																			
	复位值									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30h	ADC_SQR2	保留	规则通道序列 SQx_x 位																														
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34h	ADC_SQR3	保留	规则通道序列 SQx_x 位																														
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38h	ADC_JSQR	保留											JL [1:0]		注入通道序列 JSQx_x 位																		
	复位值												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
3Ch	ADC_JDR1	保留																JDATA[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40h	ADC_JDR2	保留																JDATA[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44h	ADC_JDR3	保留																JDATA[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48h	ADC_JDR4	保留																JDATA[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4Ch	ADC_DR	ADC2DATA[15:0]																规则DATA[15:0]															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

关于寄存器的起始地址，请参见表 1。

11 DMA 控制器

11.1 DMA 简介

直接存储器存取(DMA)用来提供在外设和存储器之间或者存储器和存储器之间的高速数据传输。无须 CPU 干预，数据可以通过 DMA 快速地移动，这就节省了 CPU 的资源来做其他操作。

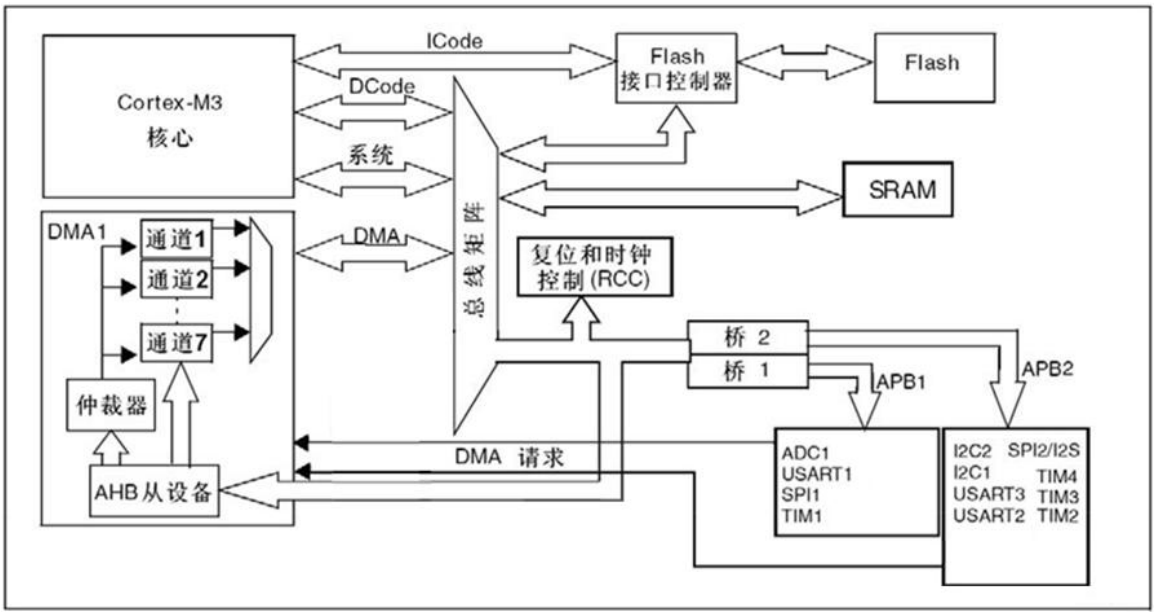
DMA 控制器有 7 个通道，每个通道专门用来管理来自于一个或多个外设对存储器访问的请求。还有一个仲裁器来协调各个 DMA 请求的优先权。

11.2 DMA 主要特性

- 7 个独立的可配置的通道(请求)。
- 每个通道都直接连接专用的硬件 DMA 请求，每个通道都同样支持软件触发。这些功能通过软件来配置。
- 在同一个 DMA 模块上，多个请求间的优先权可以通过软件编程设置(共有四级：很高、高、中等和低)，优先权设置相等时由硬件决定(请求 0 优先于请求 1，依此类推)。
- 独立数据源和目标数据区的传输宽度(字节、半字、全字)，模拟打包和拆包的过程。源和目标地址必须按数据传输宽度对齐。
- 支持循环的缓冲器管理
- 每个通道都有 3 个事件标志(DMA 半传输、DMA 传输完成和 DMA 传输出错)，这 3 个事件标志逻辑或成为一个单独的中断请求。
- 存储器和存储器间的传输
- 外设和存储器、存储器和外设之间的传输。
- 闪存、SRAM、外设的 SRAM、APB1、APB2 和 AHB 外设均可作为访问的源和目标。
- 可编程的数据传输数目：最大为 65535

下面为功能框图：

图 35 DMA 框图



11.3 功能描述

DMA 控制器和 Cortex™-M3 核心共享系统数据总线，执行直接存储器数据传输。当 CPU 和 DMA 同时访问相同的目标(RAM 或外设)时，DMA 请求会暂停 CPU 访问系统总线达若干个周期，总线仲裁器执行循环调度，以保证 CPU 至少可以得到一半的系统总线(存储器或外设)带宽。

11.3.1 DMA 处理

在发生一个事件后，外设向 DMA 控制器发送一个请求信号。DMA 控制器根据通道的优先权处理请求。当 DMA 控制器开始访问发出请求的外设时，DMA 控制器立即发送给它一个应答信号。当从 DMA 控制器得到应答信号时，外设立即释放它的请求。一旦外设释放了这个请求，DMA 控制器同时撤销应答信号。如果有更多的请求时，外设可以启动下一个周期。

总之，每次 DMA 传送由 3 个操作组成：

- 从外设数据寄存器或者从当前外设/存储器地址寄存器指示的存储器地址取数据，第一次传输时的开始地址是 DMA_CPARx 或 DMA_CMARx 寄存器指定的外设基地址或存储器单元。
- 存数据到外设数据寄存器或者当前外设/存储器地址寄存器指示的存储器地址，第一次传输时的开始地址是 DMA_CPARx 或 DMA_CMARx 寄存器指定的外设基地址或存储器单元。
- 执行一次 DMA_CNDTRx 寄存器的递减操作，该寄存器包含未完成的操作数目。

11.3.2 仲裁器

仲裁器根据通道请求的优先级来启动外设/存储器的访问。

优先权管理分 2 个阶段：

- 软件：每个通道的优先权可以在 DMA_CCRx 寄存器中设置，有 4 个等级：
  - 最高优先级
  - 高优先级
  - 中等优先级
  - 低优先级
- 硬件：如果 2 个请求有相同的软件优先级，则较低编号的通道比较高编号的通道有较高的优先权。举个例子，通道 2 优先于通道 4。

11.3.3 DMA 通道

每个通道都可以在有固定地址的外设寄存器和存储器地址之间执行 DMA 传输。DMA 传输的数据量是可编程的，最大达到 65535。包含要传输的数据项数量的寄存器，在每次传输后递减。

可编程的数据量

外设和存储器的传输数据量可以通过 DMA_CCRx 寄存器中的 PSIZE 和 MSIZE 位编程。

指针增量

通过设置 DMA_CCRx 寄存器中的 PINC 和 MINC 标志位，外设和存储器的指针在每次传输后可以有选择地完成自动增量。当设置为增量模式时，下一个要传输的地址将是前一个地址加上增量值，增量值取决与所选的数据宽度为 1、2 或 4。第一个传输的地址是存放在 DMA_CPARx/DMA_CMARx 寄存器中地址。在传输过程中，这些寄存器保持它们初始的数值，软件不能改变和读出当前正在传输的地址(它在内部的当前外设/存储器地址寄存器中)。

当通道配置为非循环模式时，传输结束后(即传输计数变为 0)将不再产生 DMA 操作。要开始新的 DMA 传输，需要在关闭 DMA 通道的情况下，在 DMA_CNDTRx 寄存器中重新写入传输数目。

在循环模式下，最后一次传输结束时，DMA_CNDTRx 寄存器的内容会自动地被重新加载为其初始数值，内部的当前外设/存储器地址寄存器也被重新加载为 DMA_CPARx/DMA_CMARx 寄存器设定的初始基地址。

通道配置过程

下面是配置 DMA 通道 x 的过程(x 代表通道号)：

1. 在 DMA_CPARx 寄存器中设置外设寄存器的地址。发生外设数据传输请求时，这个地址将是数据传输的源或目标。
2. 在 DMA_CMARx 寄存器中设置数据存储器的地址。发生外设数据传输请求时，传输的数据将从这个地址读出或写入这个地址。
3. 在 DMA_CNDTRx 寄存器中设置要传输的数据量。在每个数据传输后，这个数值递减。
4. 在 DMA_CCRx 寄存器的 PL[1:0]位中设置通道的优先级。
5. 在 DMA_CCRx 寄存器中设置数据传输的方向、循环模式、外设和存储器的增量模式、外设和存储器的数据宽度、传输一半产生中断或传输完成产生中断。
6. 设置 DMA_CCRx 寄存器的 ENABLE 位，启动该通道。

一旦启动了 DMA 通道，它既可响应连到该通道上的外设的 DMA 请求。

当传输一半的数据后，半传输标志(HTIF)被置 1，当设置了允许半传输中断位(HTIE)时，将产生一个中断请求。在数据传输结束后，传输完成标志(TCIF)被置 1，当设置了允许传输完成中断位(TCIE)时，将产生一个中断请求。

循环模式

循环模式用于处理循环缓冲区和连续的数据传输(如 ADC 的扫描模式)。在 DMA_CCRx 寄存器中的 CIRC 位用于开启这一功能。当启动了循环模式，数据传输的数目变为 0 时，将会自动地被恢复成配置通道时设置的初值，DMA 操作将会继续进行。

存储器到存储器模式

DMA 通道的操作可以在没有外设请求的情况下进行，这种操作就是存储器到存储器模式。

当设置了 DMA_CCRx 寄存器中的 MEM2MEM 位之后，在软件设置了 DMA_CCRx 寄存器中的 EN 位启动 DMA 通道时，DMA 传输将马上开始。当 DMA_CNDTRx 寄存器变为 0 时，DMA 传输结束。存储器到存储器模式不能与循环模式同时使用。

11.3.4 可编程的数据传输宽度、对齐方式和数据大小端

当 PSIZE 和 MSIZE 不相同时，DMA 模块按照下表进行数据对齐。

表 47 可编程的数据传输宽度和大小端操作(当 PINC = MINC = 1)

源端宽度	目标宽度	传输数目	源：地址/数据	传输操作	目标：地址/数据
8	8	4	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3	1: 在 0x0 读 B0[7:0]，在 0x0 写 B0[7:0] 2: 在 0x1 读 B1[7:0]，在 0x1 写 B1[7:0] 3: 在 0x2 读 B2[7:0]，在 0x2 写 B2[7:0] 4: 在 0x3 读 B3[7:0]，在 0x3 写 B3[7:0]	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3
8	16	4	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3	1: 在 0x0 读 B0[7:0]，在 0x0 写 00B0[15:0] 2: 在 0x1 读 B1[7:0]，在 0x2 写 00B1[15:0] 3: 在 0x2 读 B2[7:0]，在 0x4 写 00B2[15:0] 4: 在 0x3 读 B3[7:0]，在 0x6 写 00B3[15:0]	0x0 / 00B0 0x2 / 00B1 0x4 / 00B2 0x6 / 00B3
8	32	4	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3	1: 在 0x0 读 B0[7:0]，在 0x0 写 000000B0[31:0] 2: 在 0x1 读 B1[7:0]，在 0x4 写 000000B1[31:0] 3: 在 0x2 读 B2[7:0]，在 0x8 写 000000B2[31:0] 4: 在 0x3 读 B3[7:0]，在 0xC 写 000000B3[31:0]	0x0 / 000000B0 0x4 / 000000B1 0x8 / 000000B2 0xC / 000000B3
16	8	4	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6	1: 在 0x0 读 B1B0[15:0]，在 0x0 写 B0[7:0] 2: 在 0x2 读 B3B2[15:0]，在 0x1 写 B2[7:0] 3: 在 0x4 读 B5B4[15:0]，在 0x2 写 B4[7:0] 4: 在 0x6 读 B7B6[15:0]，在 0x3 写 B6[7:0]	0x0 / B0 0x1 / B2 0x2 / B4 0x3 / B6
16	16	4	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6	1: 在 0x0 读 B1B0[15:0]，在 0x0 写 B1B0[15:0] 2: 在 0x2 读 B3B2[15:0]，在 0x2 写 B3B2[15:0] 3: 在 0x4 读 B5B4[15:0]，在 0x4 写 B5B4[15:0] 4: 在 0x6 读 B7B6[15:0]，在 0x6 写 B7B6[15:0]	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6
16	32	4	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6	1: 在 0x0 读 B1B0[15:0]，在 0x0 写 0000B1B0[31:0] 2: 在 0x2 读 B3B2[15:0]，在 0x4 写 0000B3B2[31:0] 3: 在 0x4 读 B5B4[15:0]，在 0x8 写 0000B5B4[31:0] 4: 在 0x6 读 B7B6[15:0]，在 0xC 写 0000B7B6[31:0]	0x0 / 0000B1B0 0x4 / 0000B3B2 0x8 / 0000B5B4 0xC / 0000B7B6
32	8	4	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC	1: 在 0x0 读 B3B2B1B0[31:0]，在 0x0 写 B0[7:0] 2: 在 0x4 读 B7B6B5B4[31:0]，在 0x1 写 B4[7:0] 3: 在 0x8 读 BBBAB9B8[31:0]，在 0x2 写 B8[7:0] 4: 在 0xC 读 BFBEBDBC[31:0]，在 0x3 写 BC[7:0]	0x0 / B0 0x1 / B4 0x2 / B8 0x3 / BC
32	16	4	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC	1: 在 0x0 读 B3B2B1B0[31:0]，在 0x0 写 B1B0[15:0] 2: 在 0x4 读 B7B6B5B4[31:0]，在 0x2 写 B5B4[15:0] 3: 在 0x8 读 BBBAB9B8[31:0]，在 0x4 写 B9B8[15:0] 4: 在 0xC 读 BFBEBDBC[31:0]，在 0x6 写 BDBC[15:0]	0x0 / B1B0 0x2 / B5B4 0x4 / B9B8 0x6 / BDBC
32	32	4	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC	1: 在 0x0 读 B3B2B1B0[31:0]，在 0x0 写 B3B2B1B0[31:0] 2: 在 0x4 读 B7B6B5B4[31:0]，在 0x4 写 B7B6B5B4[31:0] 3: 在 0x8 读 BBBAB9B8[31:0]，在 0x8 写 BBBAB9B8[31:0] 4: 在 0xC 读 BFBEBDBC[31:0]，在 0xC 写 BFBEBDBC[31:0]	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC

操作一个不支持字节或半字写的 AHB 设备

当 DMA 模块开始一个 AHB 的字节或半字写操作时，数据将在 HWDATA[31:0]总线中未使用的部分重复。因此，如果 DMA 以字节或半字写入不支持字节或半字写操作的 AHB 设备时(即 HSIZE 不适于该模块)，不会发生错误，DMA 将按照下面两个例子写入 32 位 HWDATA 数据：

- 当 HSIZE=半字时，写入半字'0xABCD'，DMA 将设置 HWDATA 总线为'0xABCDABCD'。
- 当 HSIZE=字节时，写入字节'0xAB'，DMA 将设置 HWDATA 总线为'0xABABABAB'。

假定 AHB/APB 桥是一个 AHB 的 32 位从设备，它不处理 HSIZE 参数，它将按照下述方式把任何 AHB 上的字节或半字按 32 位传送到 APB 上：

- 一个 AHB 上对地址 0x0(或 0x1、0x2 或 0x3)的写字节数据'0xB0'操作，将转换到 APB 上对地址 0x0 的写字数据'0xB0B0B0B0'操作。
- 一个 AHB 上对地址 0x0(或 0x2)的写半字数据'0xB1B0'操作，将转换到 APB 上对地址 0x0 的写字数据'0xB1B0B1B0'操作。

例如，如果要写入 APB 后备寄存器(与 32 位地址对齐的 16 位寄存器)，需要配置存储器数据源宽度(MSIZE)为'16 位'，外设目标数据宽度(PSIZE)为'32 位'。

11.3.5 错误管理

读写一个保留的地址区域，将会产生 DMA 传输错误。当在 DMA 读写操作时发生 DMA 传输错误时，硬件会自动地清除发生错误的通道所对应的通道配置寄存器(DMA_CCRx)的 EN 位，该通道操作被停止。此时，在 DMA_IFR 寄存器中对应该通道的传输错误中断标志位(TEIF)将被置位，如果在 DMA_CCRx 寄存器中设置了传输错误中断允许位，则将产生中断。

11.3.6 中断

每个 DMA 通道都可以在 DMA 传输过半、传输完成和传输错误时产生中断。为应用的灵活性考虑，通过设置寄存器的不同位来打开这些中断。

表 48 DMA 中断请求

中断事件	事件标志位	使能控制位
传输过半	HTIF	HTIE
传输完成	TCIF	TCIE
传输错误	TEIF	TEIE

注意： *DMA 通道都有自己的中断向量。*

11.3.7 DMA 请求映像

DMA1 控制器

从外设(TIMx[x=1、2、3、4]、ADC1、SPI1、SPI2、I2Cx[x=1、2]和 USARTx[x=1、2、3]) 产生的 7 个请求，通过逻辑或输入到 DMA1 控制器，这意味着同时只能有一个请求有效。参见下图的 DMA1 请求映像。

外设的 DMA 请求，可以通过设置相应外设寄存器中的控制位，被独立地开启或关闭。

图 36 DMA1 请求映像

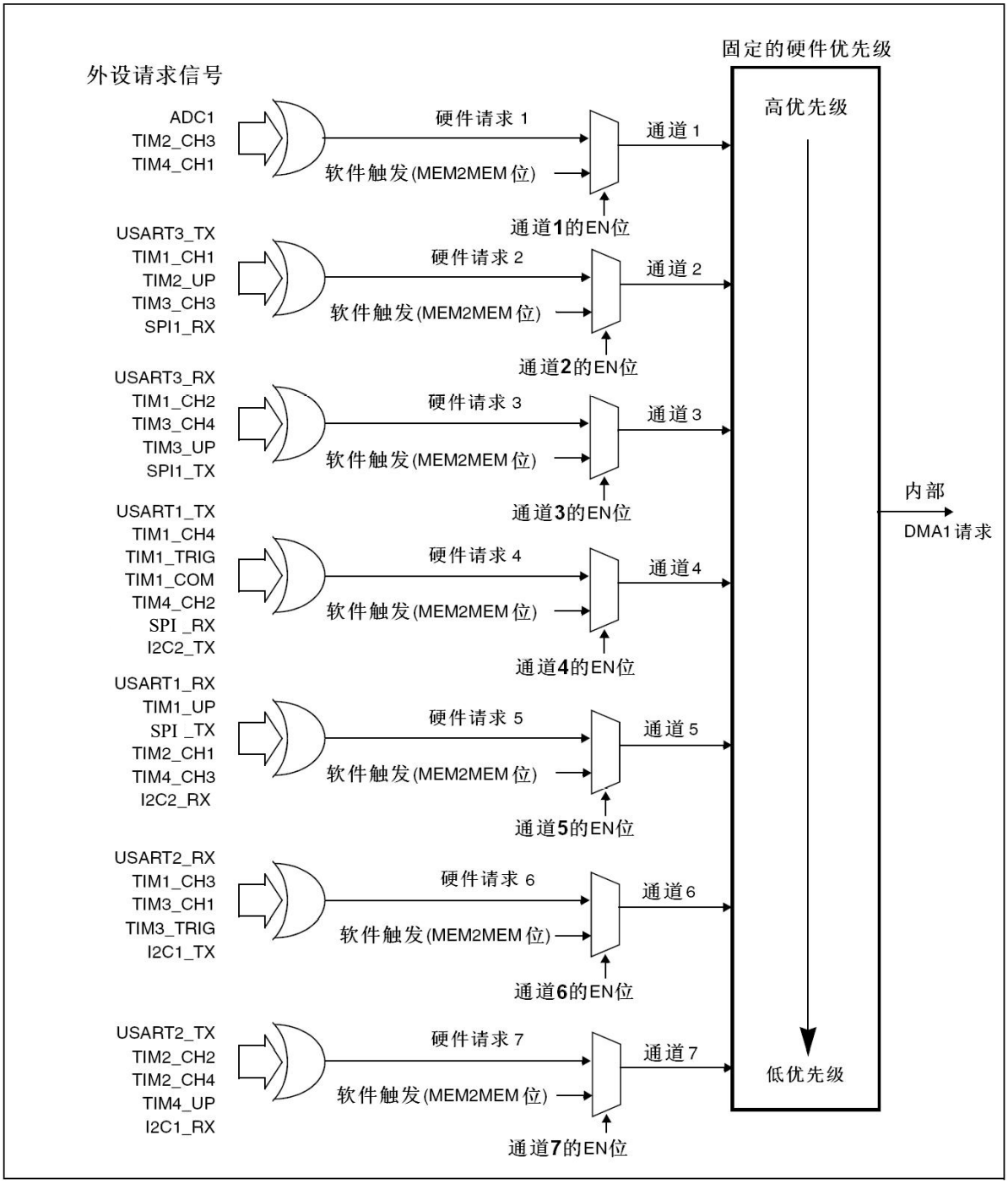

固定的硬件优先级
外设请求信号
高优先级
ADC1
TIM2_CH3
TIM4_CH1
硬件请求 1
软件触发(MEM2MEM 位)
通道 1
通道1的EN位
USART3_TX
TIM1_CH1
TIM2_UP
TIM3_CH3
SPI1_RX
硬件请求 2
软件触发(MEM2MEM 位)
通道 2
通道2的EN位
USART3_RX
TIM1_CH2
TIM3_CH4
TIM3_UP
SPI1_TX
硬件请求 3
软件触发(MEM2MEM 位)
通道 3
通道3的EN位
内部
DMA1 请求
USART1_TX
TIM1_CH4
TIM1_TRIG
TIM1_COM
TIM4_CH2
SPI _RX
I2C2_TX
硬件请求 4
软件触发(MEM2MEM 位)
通道 4
通道4的EN位
USART1_RX
TIM1_UP
SPI _TX
TIM2_CH1
TIM4_CH3
I2C2_RX
硬件请求 5
软件触发(MEM2MEM 位)
通道 5
通道5的EN位
USART2_RX
TIM1_CH3
TIM3_CH1
TIM3_TRIG
I2C1_TX
硬件请求 6
软件触发(MEM2MEM 位)
通道 6
通道6的EN位
USART2_TX
TIM2_CH2
TIM2_CH4
TIM4_UP
I2C1_RX
硬件请求 7
软件触发(MEM2MEM 位)
通道 7
通道7的EN位
低优先级

表 49 各个通道的 DMA1 请求一览

外设	通道 1	通道 2	通道 3	通道 4	通道 5	通道 6	通道 7
ADC1	ADC1						
SPI		SPI1_RX	SPI1_TX	SPI2_RX	SPI2_TX		
USART		USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I^2C				I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1		TIM1_CH1	TIM1_CH2	TIM1_TX4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	
TIM2	TIM2_CH3	TIM2_UP			TIM2_CH1		TIM2_CH2 TIM2_CH4
TIM3		TIM3_CH3	TIM3_CH4 TIM3_UP			TIM3_CH1 TIM3_TRIG	
TIM4	TIM4_CH1			TIM4_CH2	TIM4_CH3		TIM4_UP

11.4 DMA 寄存器

关于寄存器描述中用到的缩写，请参见第 2 章。

11.4.1 DMA 中断状态寄存器(DMA_ISR)

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
				r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:28	保留，始终读为 0。
位 27，23，19，15，11，7，3	**TEIFx**：通道 x 的传输错误标志(x = 1 … 7) (Channel x transfer error flag) 硬件设置这些位。在 DMA_IFCR 寄存器的相应位写入'1'可以清除这里对应的标志位。 0：在通道 x 没有传输错误(TE)； 1：在通道 x 发生了传输错误(TE)。
位 26，22，18，14，10，6，2	**HTIFx**：通道 x 的半传输标志(x = 1 … 7) (Channel x half transfer flag) 硬件设置这些位。在 DMA_IFCR 寄存器的相应位写入'1'可以清除这里对应的标志位。 0：在通道 x 没有半传输事件(HT)； 1：在通道 x 产生了半传输事件(HT)。
位 25，21，17，13，9，5，1	**TCIFx**：通道 x 的传输完成标志(x = 1 … 7) (Channel x transfer complete flag) 硬件设置这些位。在 DMA_IFCR 寄存器的相应位写入'1'可以清除这里对应的标志位。 0：在通道 x 没有传输完成事件(TC)； 1：在通道 x 产生了传输完成事件(TC)。

位 24，20，16，12，8，4，0	**GIFx**：通道 x 的全局中断标志(x = 1 … 7) (Channel x global interrupt flag) 硬件设置这些位。在 DMA_IFCR 寄存器的相应位写入'1'可以清除这里对应的标志位。 0：在通道 x 没有 TE、HT 或 TC 事件； 1：在通道 x 产生了 TE、HT 或 TC 事件。

11.4.2 DMA 中断标志清除寄存器(DMA_IFCR)

偏移地址：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留				CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:28	保留，始终读为 0。
位 27，23，19，15，11，7，3	**CTEIFx**：清除通道 x 的传输错误标志(x = 1 … 7) (Channel x transfer error clear) 这些位由软件设置和清除。 0：不起作用 1：清除 DMA_ISR 寄存器中的对应 TEIF 标志。
位 26，22，18，14，10，6，2	**CHTIFx**：清除通道 x 的半传输标志(x = 1 … 7) (Channel x half transfer clear) 这些位由软件设置和清除。 0：不起作用 0：清除 DMA_ISR 寄存器中的对应 HTIF 标志。
位 25，21，17，13，9，5，1	**CTCIFx**：清除通道 x 的传输完成标志(x = 1 … 7) (Channel x transfer complete clear) 这些位由软件设置和清除。 0：不起作用 0：清除 DMA_ISR 寄存器中的对应 TCIF 标志。
位 24，20，16，12，8，4，0	**CGIFx**：清除通道 x 的全局中断标志(x = 1 … 7) (Channel x global interrupt clear) 这些位由软件设置和清除。 0：不起作用 0：清除 DMA_ISR 寄存器中的对应的 GIF、TEIF、HTIF 和 TCIF 标志。

11.4.3 DMA 通道 x 配置寄存器(DMA_CCRx)(x = 1…7)

偏移地址：0x08 + 20 x (通道编号– 1)

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:15	保留，始终读为 0。
位 14	**MEM2MEM**：存储器到存储器模式(Memory to memory mode) 该位由软件设置和清除。 0：非存储器到存储器模式； 1：启动存储器到存储器模式。
位 13:12	**PL[1:0]**：通道优先级(Channel priority level) 这些位由软件设置和清除。 00：低 01：中 10：高 11：最高
位 11:10	**MSIZE[1:0]**：存储器数据宽度(Memory size) 这些位由软件设置和清除。 00：8 位 01：16 位 10：32 位 11：保留
位 9:8	**PSIZE[1:0]**：外设数据宽度(Peripheral size) 这些位由软件设置和清除。 00：8 位 01：16 位 10：32 位 11：保留
位 7	**MINC**：存储器地址增量模式(Memory increment mode) 该位由软件设置和清除。 0：不执行存储器地址增量操作 1：执行存储器地址增量操作
位 6	**PINC**：外设地址增量模式(Peripheral increment mode) 该位由软件设置和清除。 0：不执行外设地址增量操作 1：执行外设地址增量操作

位 5	**CIRC**：循环模式(Circular mode) 该位由软件设置和清除。 0：不执 行循环操作 1：执行循环操作
位 4	**DIR**：数据传输方向(Data transfer direction) 该位由软件设置和清除。 0：从外设读 1：从存储器读
位 3	**TEIE**：允许传输错误中断(Transfer error interrupt enable) 该位由软件设置和清除。 0：禁止 TE 中断 0：允许 TE 中断
位 2	**HTIE**：允许半传输中断(Half transfer interrupt enable) 该位由软件设置和清除。 0：禁止 HT 中断 0：允许 HT 中断
位 1	**TCIE**：允许传输完成中断(Transfer complete interrupt enable) 该位由软件设置和清除。 0：禁止 TC 中断 0：允许 TC 中断
位 0	**EN**：通道开启 (Channel enable) 该位由软件设置和清除。 0：通道不工作 1：通道开启

11.4.4 DMA 通道 x 传输数量寄存器(DMA_CNDTRx)(x = 1…7)

偏移地址：0x0C + 20 x (通道编号– 1)

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16	保留，始终读为 0。
位 15:0	**NDT[15:0]**：数据传输数量(Number of data to transfer) 数据传输数量为 0 至 65535。这个寄存器只能在通道不工作(DMA_CCRx 的 EN=0)时写入。通道开启后该寄存器变为只读，指示剩余的待传输字节数目。寄存器内容在每次 DMA 传输后递减。 数据传输结束后，寄存器的内容或者变为 0；或者当该通道配置为自动重加载模式时，寄存器的内容将被自动重新加载为之前配置时的数值。 当寄存器的内容为 0 时，无论通道是否开启，都不会发生任何数据传输。

11.4.5 DMA 通道 x 外设地址寄存器(DMA_CPARx)(x = 1…7)

偏移地址：0x10 + 20 x (通道编号– 1)

复位值：0x0000 0000

当开启通道(DMA_CCRx 的 EN=1)时不能写该寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA[31:0]																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0	**PA[31:0]**：外设地址(Peripheral address) 外设数据寄存器的基地址，作为数据传输的源或目标。 当 PSIZE='01'(16 位)，不使用 PA[0]位。操作自动地与半字地址对齐。 当 PSIZE='10'(32 位)，不使用 PA[1:0]位。操作自动地与字地址对齐。

11.4.6 DMA 通道 x 存储器地址寄存器(DMA_CMARx)(x = 1…7)

偏移地址：0x14 + 20 x (通道编号– 1)

复位值：0x0000 0000

当开启通道(DMA_CCRx 的 EN=1)时不能写该寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[31:0]																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0	**MA[31:0]**：存储器地址 存储器地址作为数据传输的源或目标 当 MSIZE='01'(16 位)，不使用 MA[0]位。操作自动地与半字地址对齐。 当 MSIZE='10'(32 位)，不使用 MA[1:0]位。操作自动地与字地址对齐。

11.4.7 DMA 寄存器映像

关于寄存器的起始地址，参见表 1。

表 50 DMA 寄存器映像和复位

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	DMA_ISR	保				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5	TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
	复位值					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004h	DMA_IFCR	保				CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5	CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
	复位值					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
008h	DMA_CCR1	保																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	DMA_CNDTR1	保																		0	0	0	0	NDT[15:0]									
	复位值																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	DMA_CPAR1	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	DMA_CMAR1	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	保留																																
01Ch	DMA_CCR2	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	DMA_CNDTR2	保留																		NDT[15:0]													
	复位值																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
024h	DMA_CPAR2	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028h	DMA_CMAR2	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02Ch	保留																																
030h	DMA_CCR3	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
034h	DMA_CNDTR3	保留																		NDT[15:0]													
	复位值																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
038h	DMA_CPAR3	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03Ch	DMA_CMAR3	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
040h	保留																																

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
044h	DMA_CCR4	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
048h	DMA_CNDTR4	保留																	NDT[15:0]														
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04Ch	DMA_CPAR4	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
050h	DMA_CMAR4	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
054h	保留																																
058h	DMA_CCR5	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05Ch	DMA_CNDTR5	保留																	NDT[15:0]														
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
060h	DMA_CPAR5	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
064h	DMA_CMAR5	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
068h	保留																																
06Ch	DMA_CCR6	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	S 复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
070h	DMA_CNDTR6	保留																	NDT[15:0]														
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
074h	DMA_CPAR6	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
078h	DMA_CMAR6	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07Ch	保留																																
080h	DMA_CCR7	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
084h	DMA_CNDTR7	保留																	NDT[15:0]														
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
088h	DMA_CPAR7	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08Ch	DMA_CMAR7	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
090h	保留																																

12 高级控制定时器(TIM1)

12.1 TIM1 简介

高级控制定时器(TIM1)由一个 16 位的自动装载计数器组成，它由一个可编程的预分频器 驱动。

它适合多种用途，包含测量输入信号的脉冲宽度(输入捕获)，或者产生输出波形(输出比较、PWM、嵌入死区时间的互补 PWM 等)。

使用定时器预分频器和 RCC 时钟控制预分频器，可以实现脉冲宽度和波形周期从几个微秒到几个毫秒的调节。

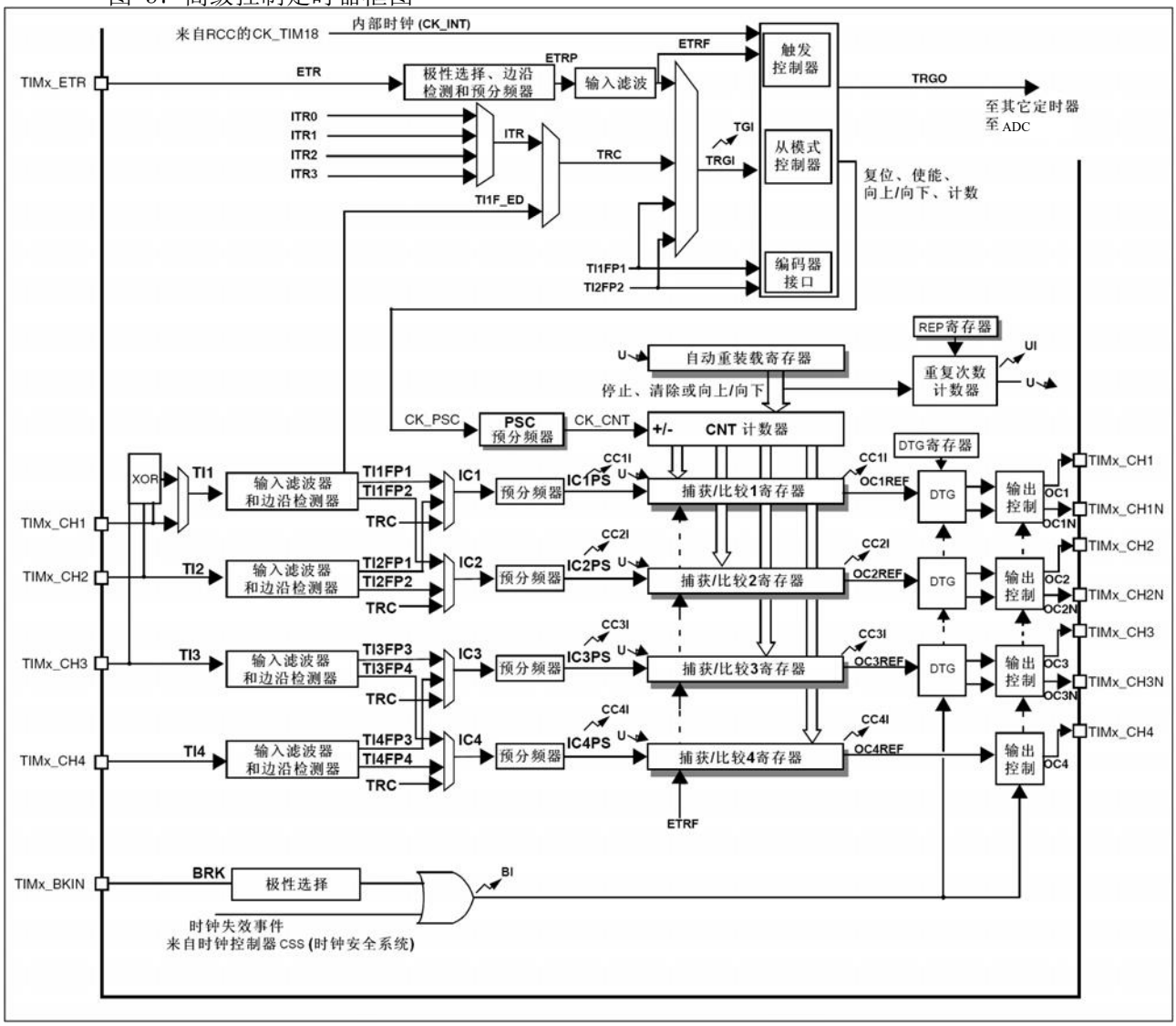
高级控制定时器(TIM1)和通用定时器(TIMx)是完全独立的，它们不共享任何资源。它们可以同步操作，具体描述参看 13.3.20 节。

12.2 TIM1 主要特征

TIM1 定时器的功能包括：

- 16 位向上、向下、向上/下自动装载计数器
- 16 位可编程(可以实时修改)预分频器，计数器时钟频率的分频系数为 1～65535 之间的任意数值
- 多达 4 个独立通道：
  - 输入捕获
  - 输出比较
  - PWM 生成(边缘或中间对齐模式)
  - 单脉冲模式输出
- 死区时间可编程的互补输出
- 使用外部信号控制定时器和定时器互联的同步电路
- 允许在指定数目的计数器周期之后更新定时器寄存器的重复计数器
- 刹车输入信号可以将定时器输出信号置于复位状态或者一个已知状态
- 如下事件发生时产生中断/DMA：
  - 更新：计数器向上溢出/向下溢出，计数器初始化(通过软件或者内部/外部触发)
  - 触发事件(计数器启动、停止、初始化或者由内部/外部触发计数)
  - 输入捕获
  - 输出比较
  - 刹车信号输入
- 支持针对定位的增量(正交)编码器和霍尔传感器电路
- 触发输入作为外部时钟或者按周期的电流管理

图 37 高级控制定时器框图



注： Reg *根据控制位的设定，在 U(更新)事件时传送预加载寄存器的内容至工作寄存器*

事件

中断和 DMA 输出

12.3 TIM1 功能描述

12.3.1 时基单元

可编程高级控制定时器的主要部分是一个 16 位计数器和与其相关的自动装载寄存器。这个计数器可以向上计数、向下计数或者向上向下双向计数。此计数器时钟由预分频器分频得到。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写，即使计数器还在运行读写仍然有效。

时基单元包含：

- 计数器寄存器(TIMx_CNT)
- 预分频器寄存器(TIMx_PSC)
- 自动装载寄存器(TIMx_ARR)
- 重复次数寄存器(TIMx_RCR)

自动装载寄存器是预先装载的，写或读自动重装载寄存器将访问预装载寄存器。根据在TIMx_CR1 寄存器中的自动装载预装载使能位(ARPE)的设置，预装载寄存器的内容被立即或在每次的更新事件 UEV 时传送到影子寄存器。当计数器达到溢出条件(向下计数时的下溢条件)并当TIMx_CR1 寄存器中的 UDIS 位等于 0 时，产生更新事件。更新事件也可以由软件产生。随后会详 细描述每一种配置下更新事件的产生。

计数器由预分频器的时钟输出 CK_CNT 驱动，仅当设置了计数器 TIMx_CR1 寄存器中的计数器使能位(CEN)时，CK_CNT 才有效。(更多有关使能计数器的细节，请参见控制器的从模式描述)。

注意，在设置了 TIMx_CR 寄存器的 CEN 位的一个时钟周期后，计数器开始计数。

预分频器描述

预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。它是基于一个(在 TIMx_PSC 寄存器中的)16 位寄存器控制的 16 位计数器。因为这个控制寄存器带有缓冲器，它能够在运行时被改变。新的预分频器的参数在下一次更新事件到来时被采用。

图 38 和图 39 给出了在预分频器运行时，更改计数器参数的例子。

图 38 当预分频器的参数从 1 变到 2 时，计数器的时序图

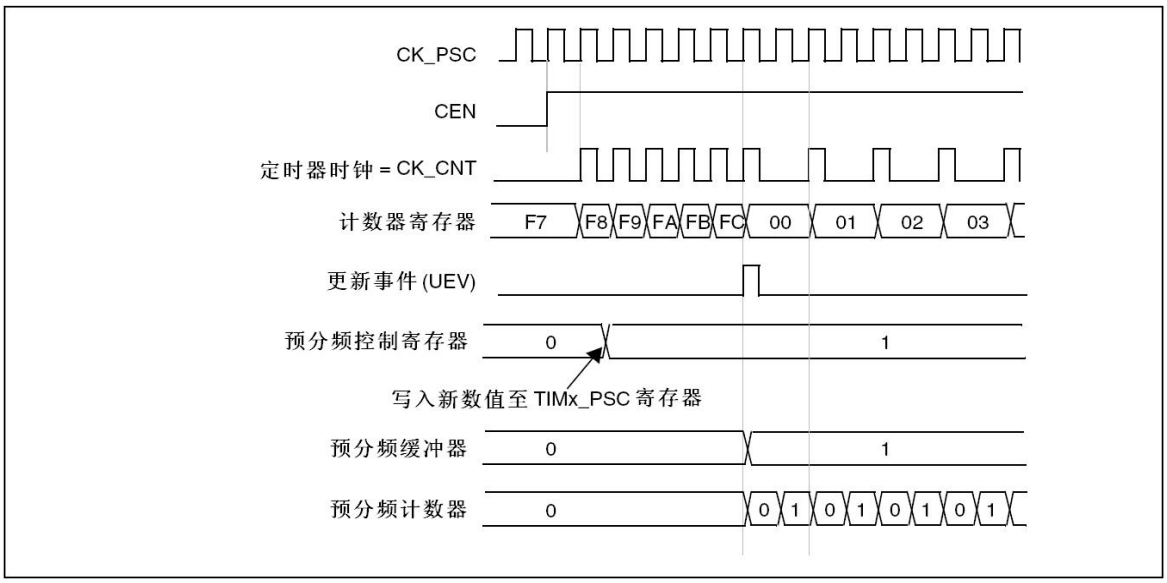
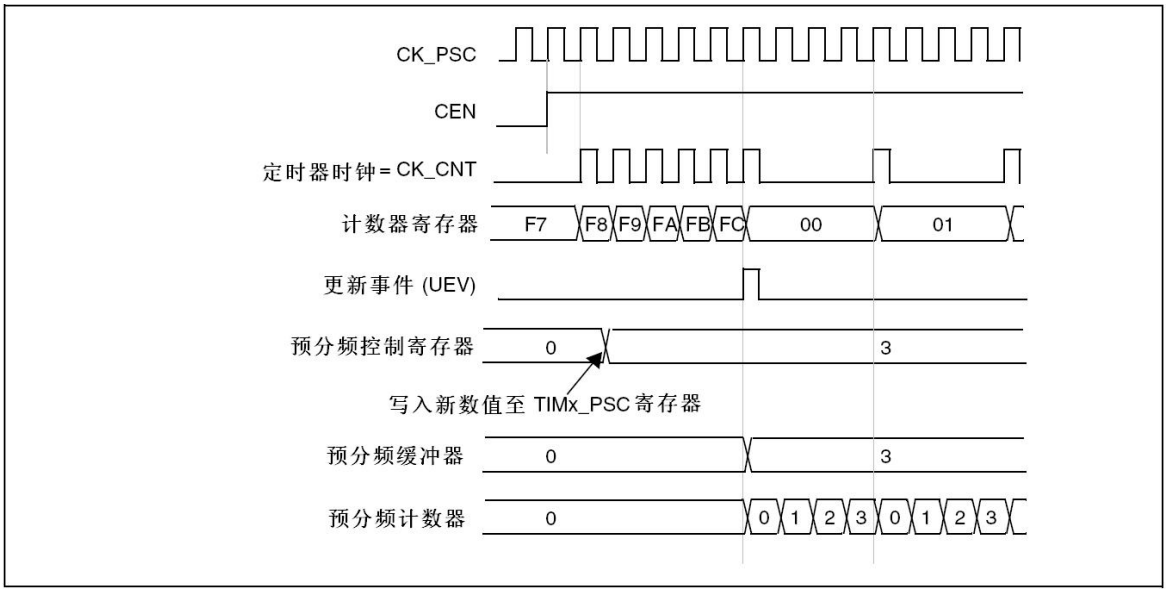


图 39 当预分频器的参数从 1 变到 4 时，计数器的时序图



12.3.2 计数器模式

向上计数模式

在向上计数模式中，计数器从 0 计数到自动加载值(TIMx_ARR 计数器的内容)，然后重新从 0 开始计数并且产生一个计数器溢出事件。

如果使用了重复计数器功能，在向上计数达到设置的重复计数次数(TIMx_RCR)时，产生更新事件(UEV)；否则每次计数器溢出时才产生更新事件。

在 TIMx_EGR 寄存器中(通过软件方式或者使用从模式控制器)设置 UG 位也同样可以产生一个更新事件。

设置 TIMx_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清'0'之前，将不产生更新事件。但是在应该产生更新事 件时，计数器仍会被清'0'，同时预分频器的计数也被请 0(但预分频器的数值不变)。此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求)，设置 UG 位将产生一个更新事件 UEV，但硬件不设置 UIF 标志(即不产生中断或 DMA 请求)。这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

当发生一个更新事件时，所有的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIMx_SR 寄存器中的 UIF 位)。

- 重复计数器被重新加载为 TIMx_RCR 寄存器的内容。
- 自动装载影子寄存器被重新置入预装载寄存器的值(TIMx_ARR)。
- 预分频器的缓冲区被置入预装载寄存器的值(TIMx_PSC 寄存器的内容)。

下图给出一些例子，当 TIMx_ARR=0x36 时计数器在不同时钟频率下的动作。

图 40 计数器时序图，内部时钟分频因子为 1

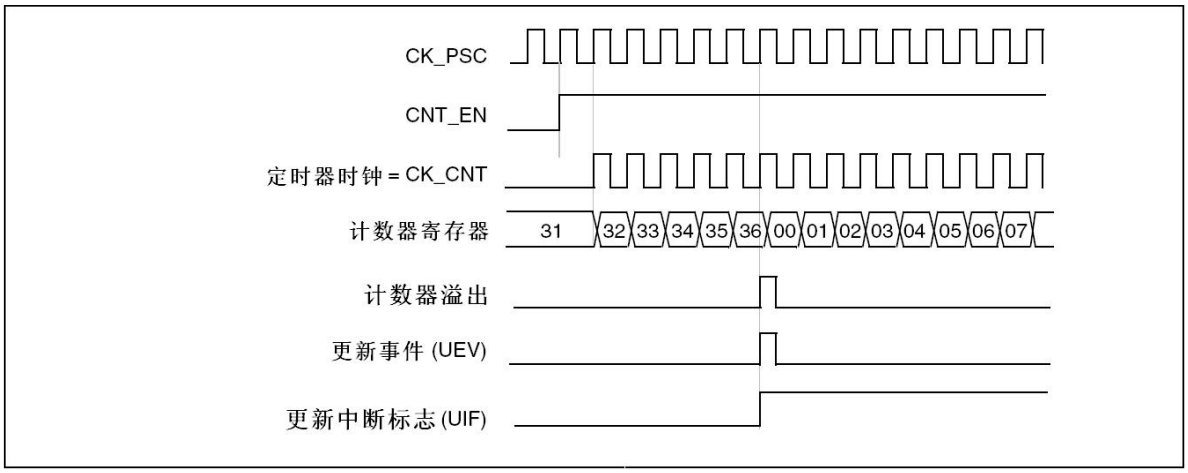


图 41 计数器时序图，内部时钟分频因子为 2

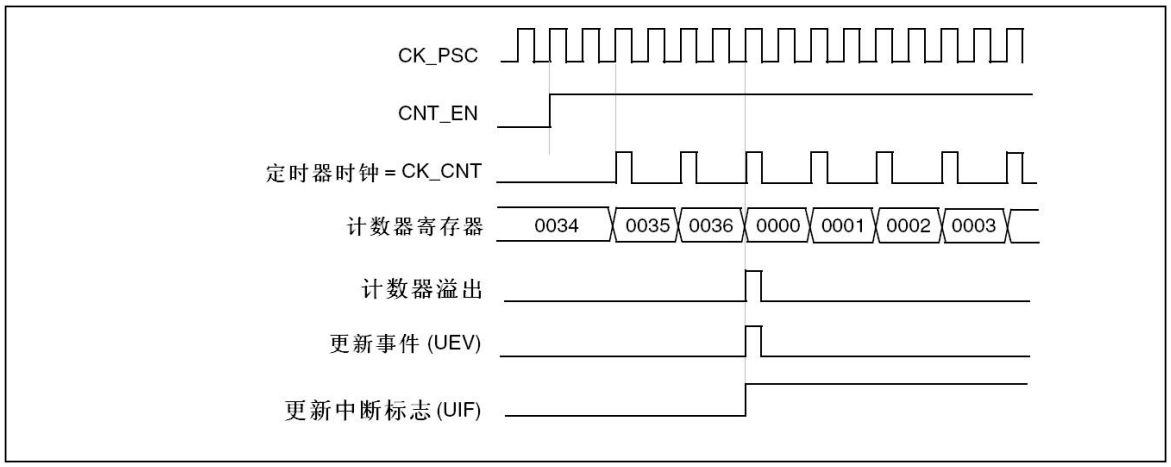


图 42 计数器时序图，内部时钟分频因子为 4

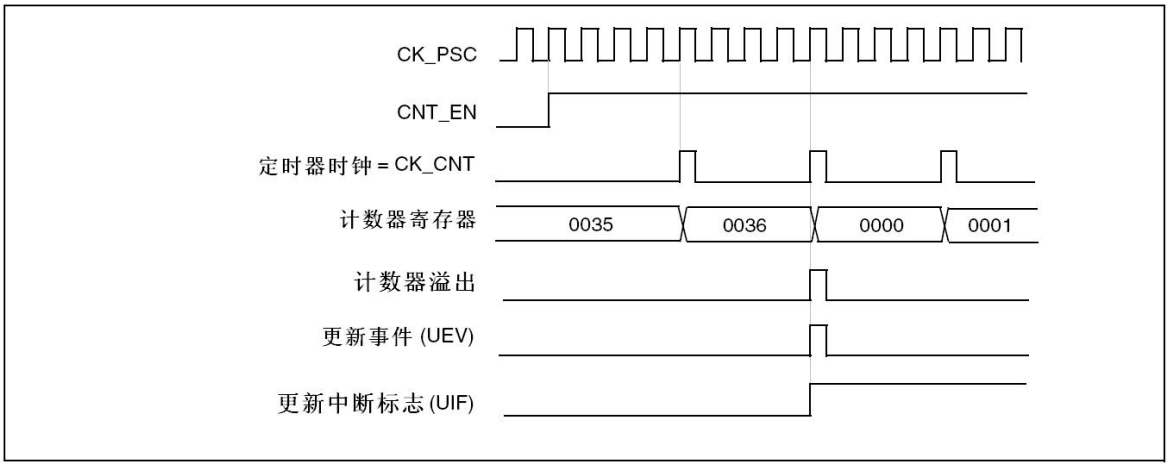


图 43 计数器时序图，内部时钟分频因子为 N

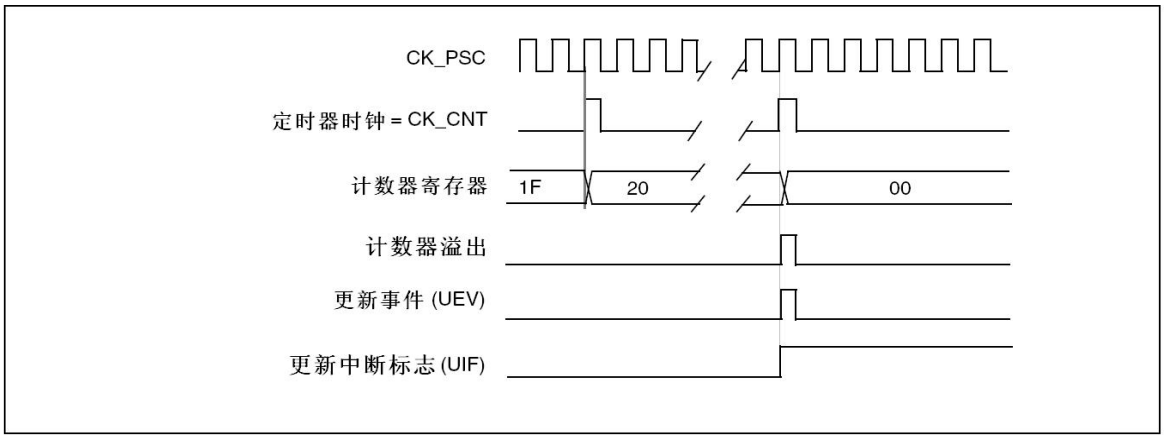


图 44 计数器时序图，当 ARPE=0 时的更新事件(TIMx_ARR 没有预装入)

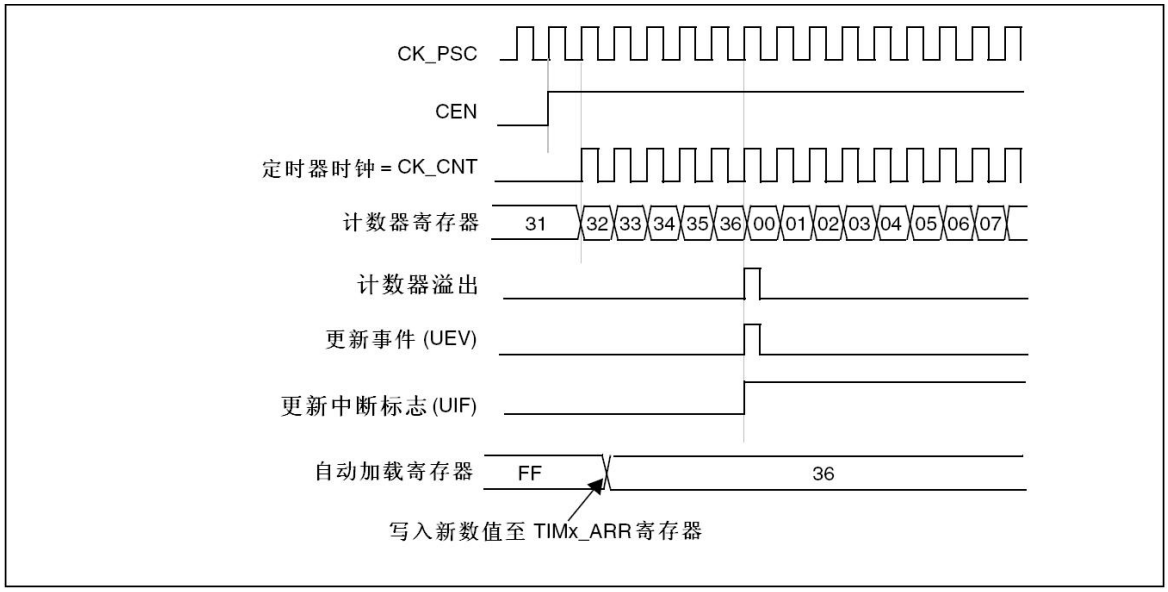
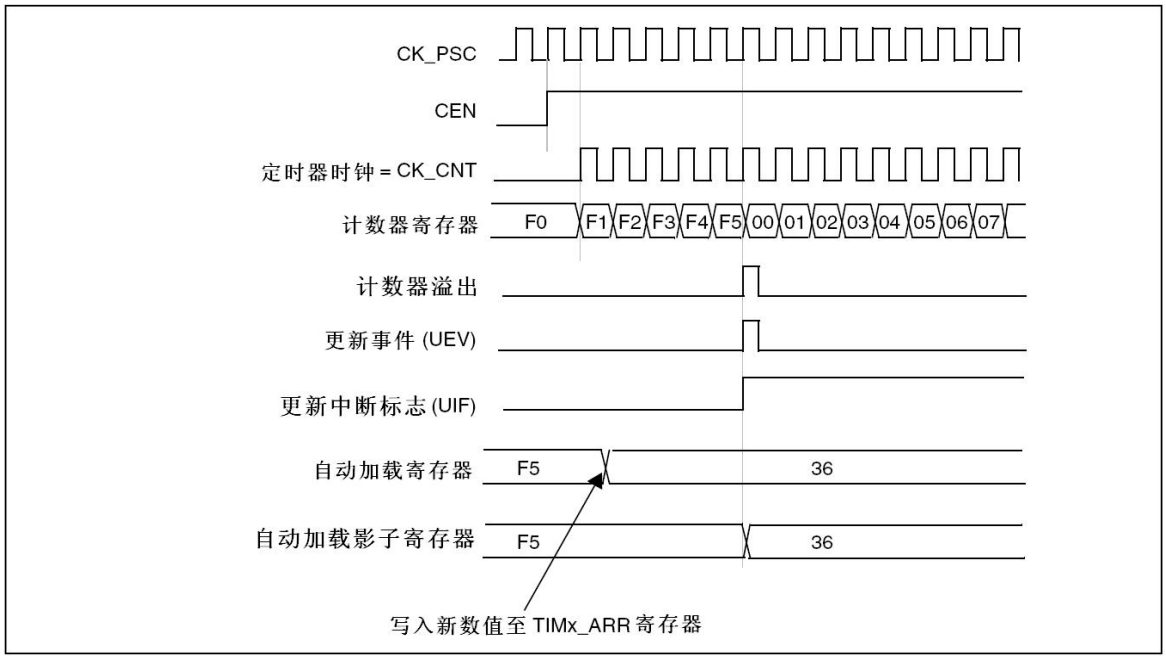


图 45 计数器时序图，当 ARPE=1 时的更新事件(预装入了 TIMx_ARR)



向下计数模式

在向下模式中，计数器从自动装入的值(TIMx_ARR 计数器的值)开始向下计数到 0，然后从自动装入的值重新开始并且产生一个计数器向下溢出事件。

如果使用了重复计数器，当向下计数重复了重复计数寄存器(TIMx_RCR)中设定的次数后，将产生更新事件(UEV)，否则每次计数器下溢时才产生更新事件。

在 TIMx_EGR 寄存器中(通过软件方式或者使用从模式控制器)设置 UG 位，也同样可以产生一个更新事件。

设置 TIMx_CR1 寄存器的 UDIS 位可以禁止 UEV 事件。这样可以避免向预装载寄存器中写入新值时更新影子寄存器。因此 UDIS 位被清为 0 之前不会产生更新事件。然而，计数器仍会从当前自动加载值重新开始计数，并且预分频器的计数器重新从 0 开始(但预分频系数不变)。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求) ，设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志(因此不产生中断和 DMA 请求)，这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且(根据 URS 位的设置)更新标志位(TIMx_SR 寄存器中的 UIF 位)也被设置。

- 重复计数器被重置为 TIMx_RCR 寄存器中的内容
- 预分频器的缓存器被加载为预装载的值(TIMx_PSC 寄存器的值)。
- 当前的自动加载寄存器被更新为预装载值(TIMx_ARR 寄存器中的内容)。注：自动装载在计数器重载入之前被更新，因此下一个周期将是预期的值。

以下是一些当 TIMx_ARR=0x36 时，计数器在不同时钟频率下的操作例子。

图 46 计数器时序图，内部时钟分频因子为 1

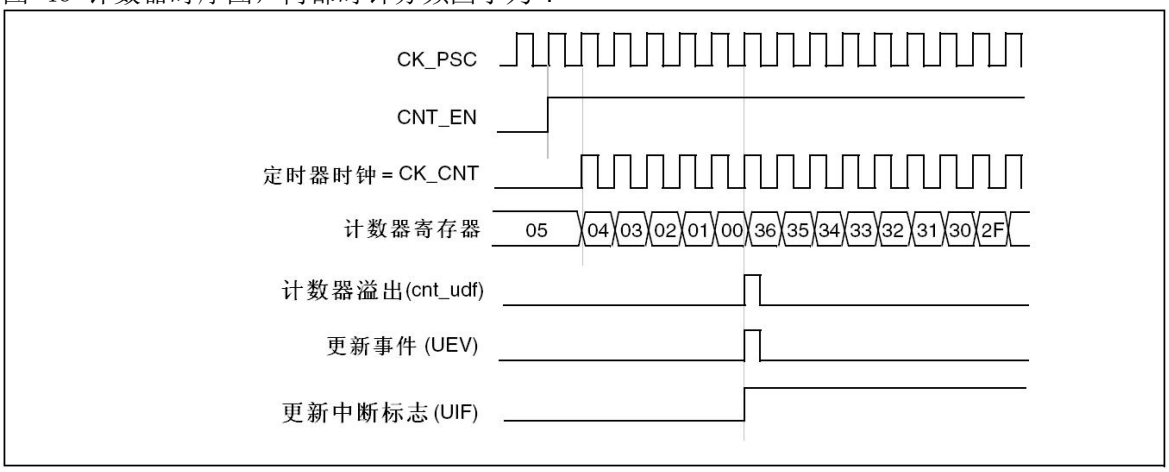


图 47 计数器时序图，内部时钟分频因子为 2

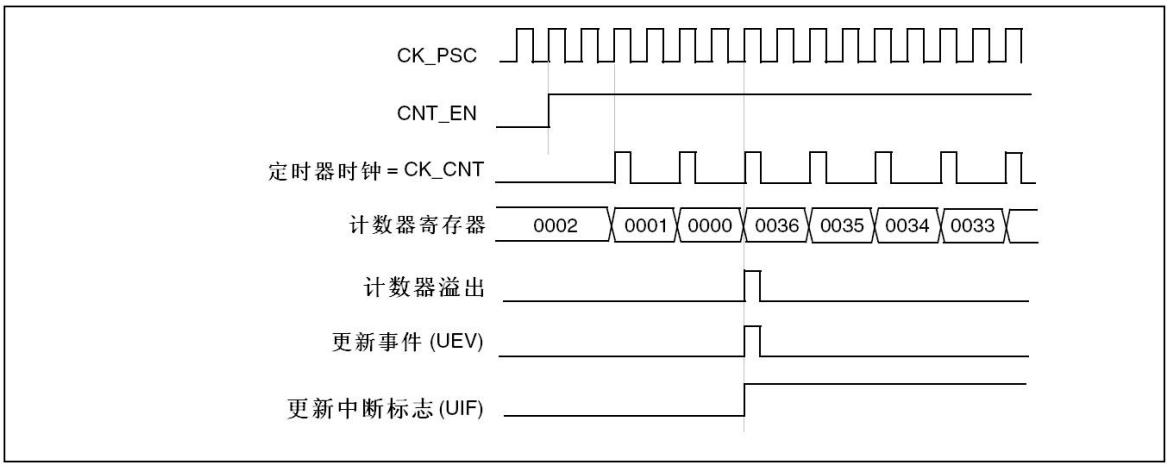


图 48 计数器时序图，内部时钟分频因子为 4

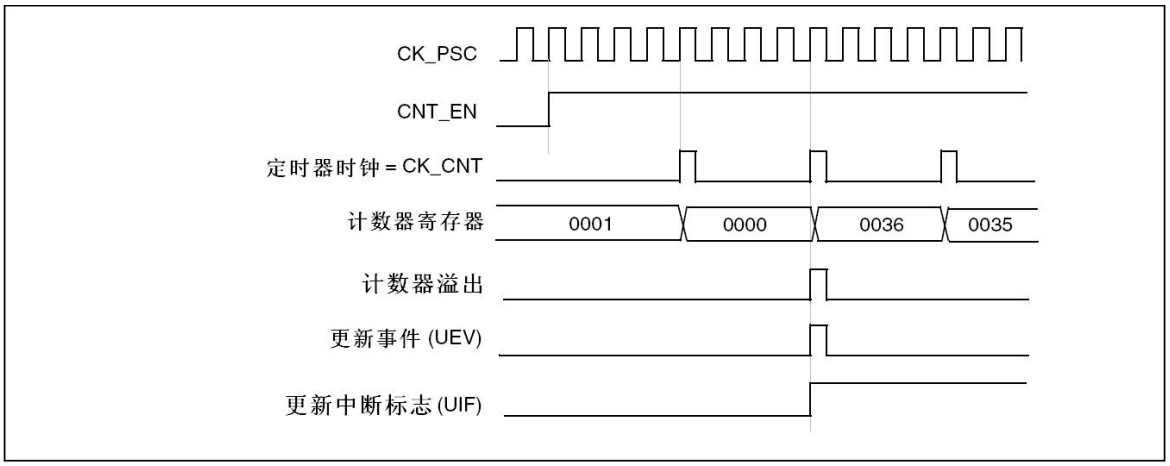


图 49 计数器时序图，内部时钟分频因子为 N

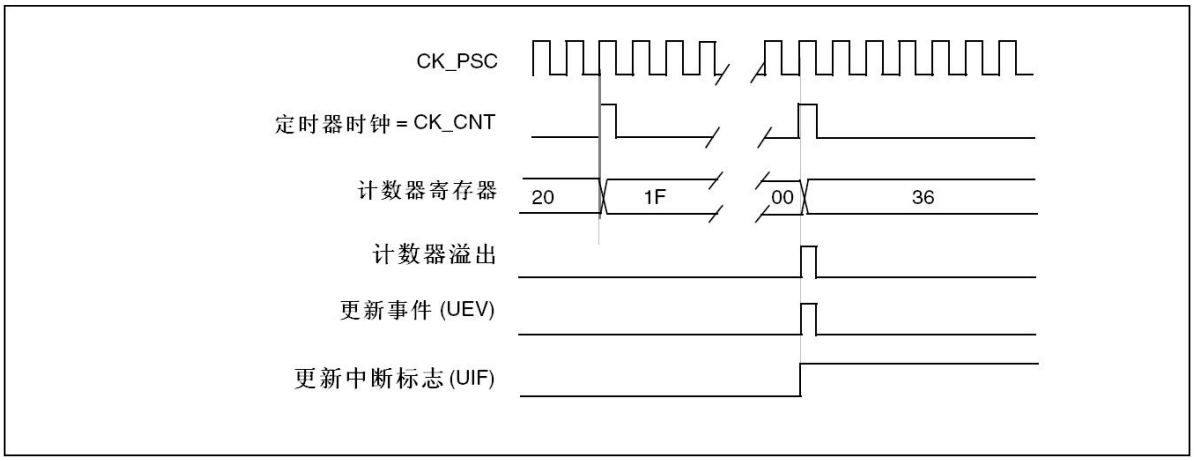
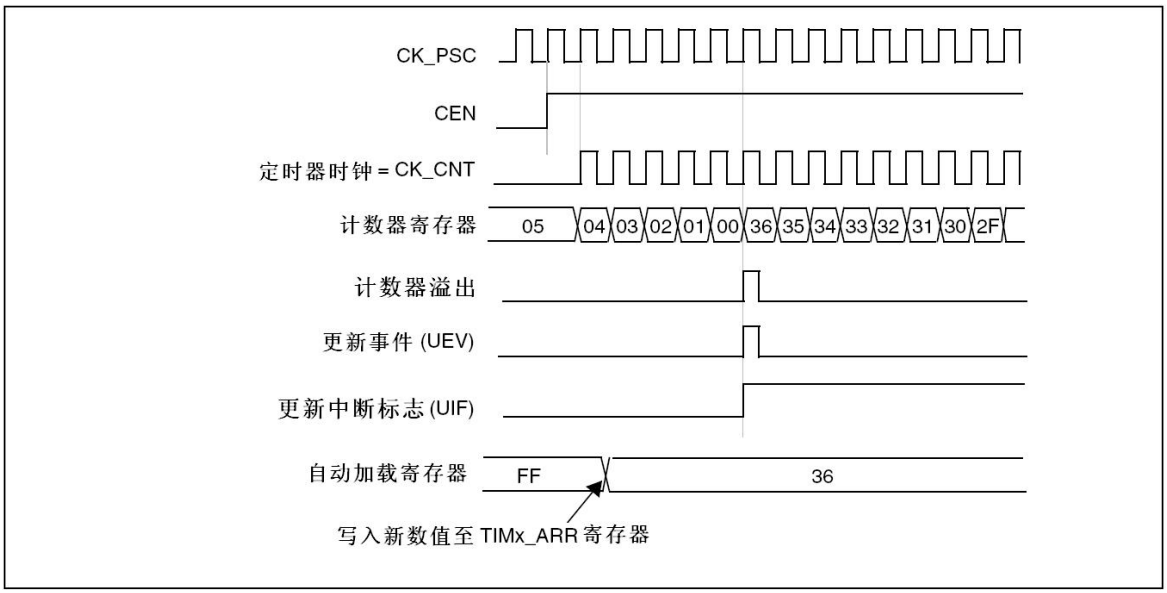


图 50 计数器时序图，当没有使用重复计数器时的更新事件



中央对齐模式(向上/向下计数)

在中央对齐模式，计数器从 0 开始计数到自动加载的值(TIMx_ARR 寄存器)−1，产生一个计数器溢出事件，然后向下计数到 1 并且产生一个计数器下溢事件；然后再从 0 开始重新计数。

在此模式下，不能写入 TIMx_CR1 中的 DIR 方向位。它由硬件更新并指示当前的计数方向。

可以在每次计数上溢和每次计数下溢时产生更新事件；也可以通过(软件或者使用从模式控制器)设置 TIMx_EGR 寄存器中的 UG 位产生更新事件。然后，计数器重新从 0 开始计数，预分频器也重新从 0 开始计数。

设置 TIMx_CR1 寄存器中的 UDIS 位可以禁止 UEV 事件。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。因此 UDIS 位被清为 0 之前不会产生更新事件。然而，计数器仍会根据当前自动重加载的值，继续向上或向下计数。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求) ，设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志(因此不产生中断和 DMA 请求)，这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

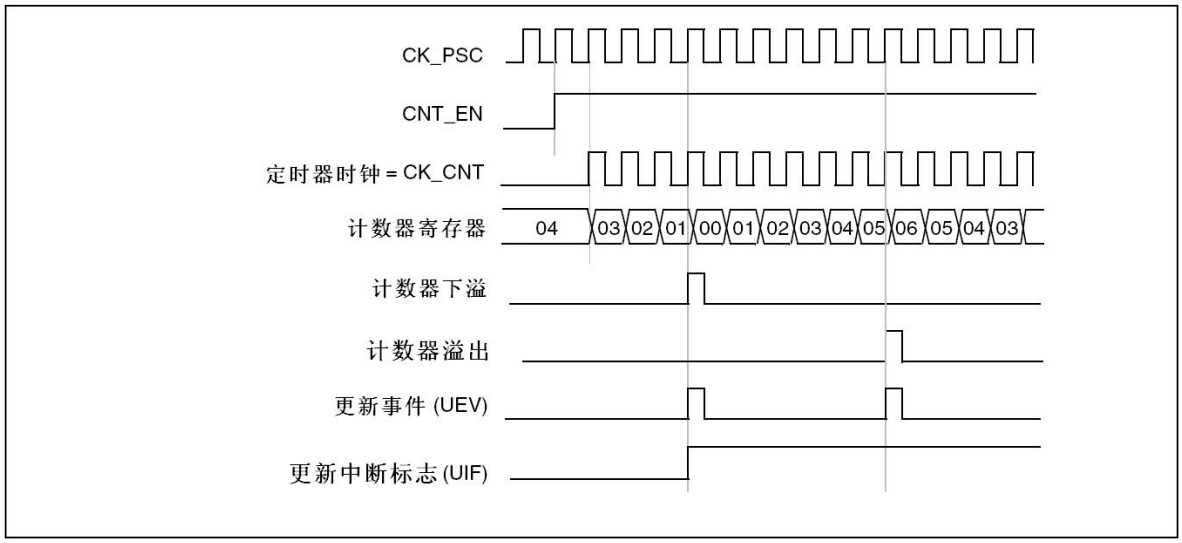
当发生更新事件时，所有的寄存器都被更新，并且(根据 URS 位的设置)更新标志位(TIMx_SR 寄存器中的 UIF 位)也被设置。

- 重复计数器被重置为 TIMx_RCR 寄存器中的内容
- 预分频器的缓存器被加载为预装载(TIMx_PSC 寄存器)的值。

- 当前的自动加载寄存器被更新为预装载值(TIMx_ARR 寄存器中的内容)。注：如果因为计数器溢出而产生更新，自动重装载将在计数器重载入之前被更新，因此下一个周期将是预期 的值(计数器被装载为新的值)。

以下是一些计数器在不同时钟频率下的操作的例子：

图 51 计数器时序图，内部时钟分频因子为 1，TIMx_ARR=0x6



1．这里使用了中心对齐模式 1(详见 13.4.1 节)。

图 52 计数器时序图，内部时钟分频因子为 2

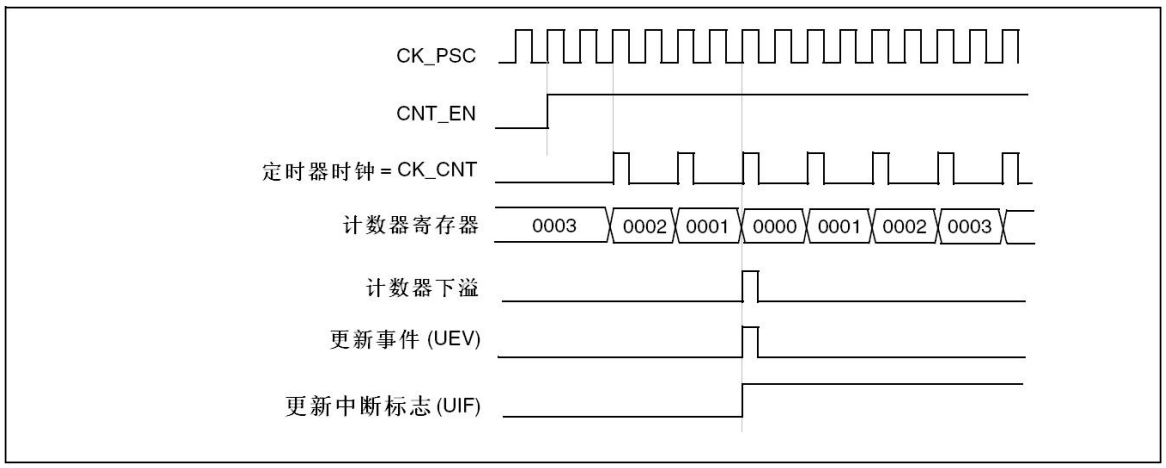


图 53 计数器时序图，内部时钟分频因子为 4，TIMx_ARR=0x36

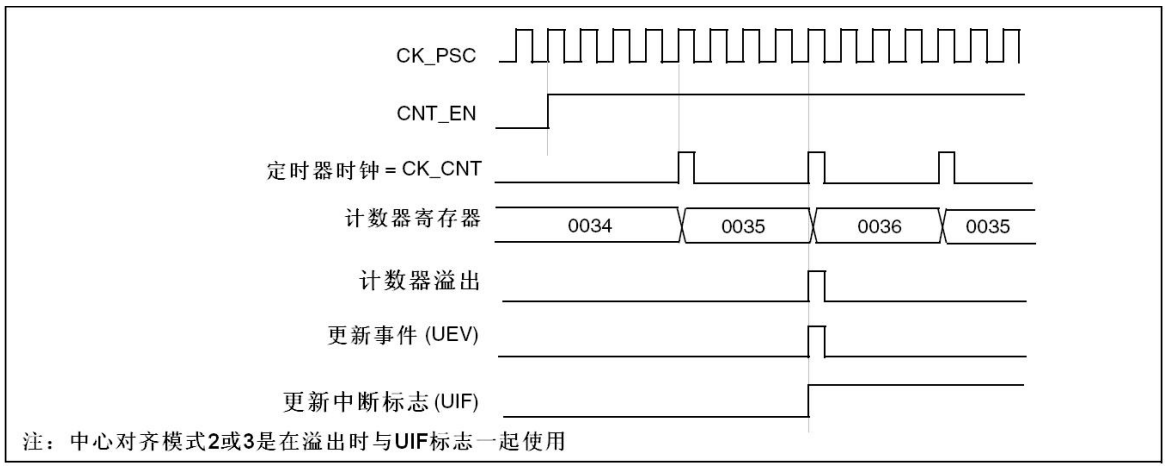


图 54 计数器时序图，内部时钟分频因子为 N

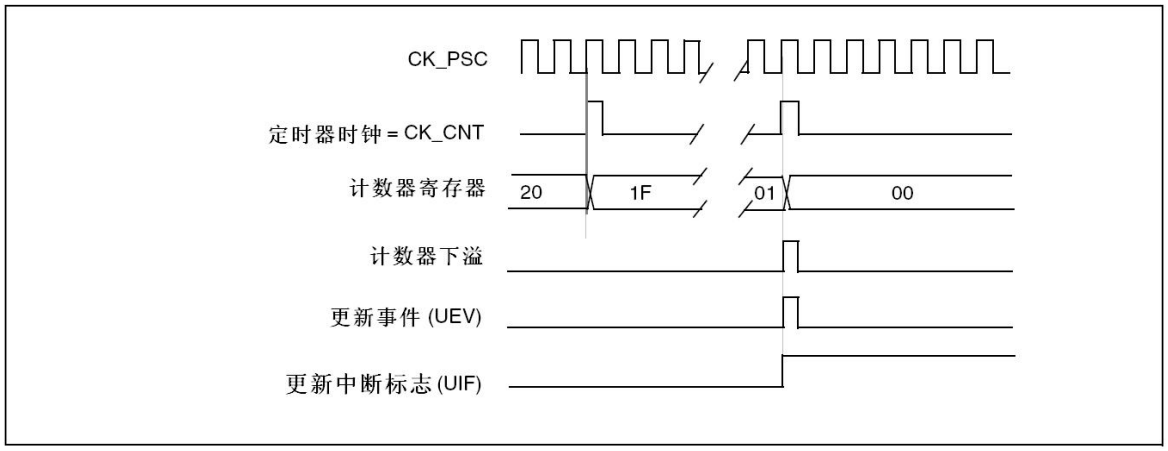


图 55 计数器时序图，ARPE=1 时的更新事件(计数器下溢)

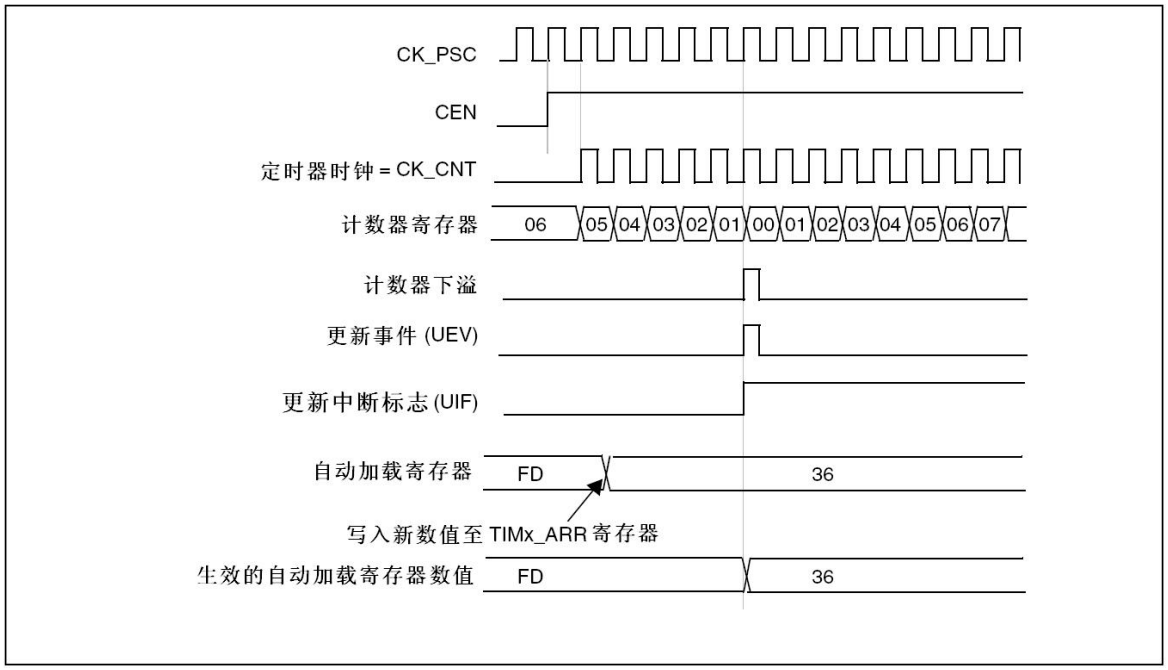
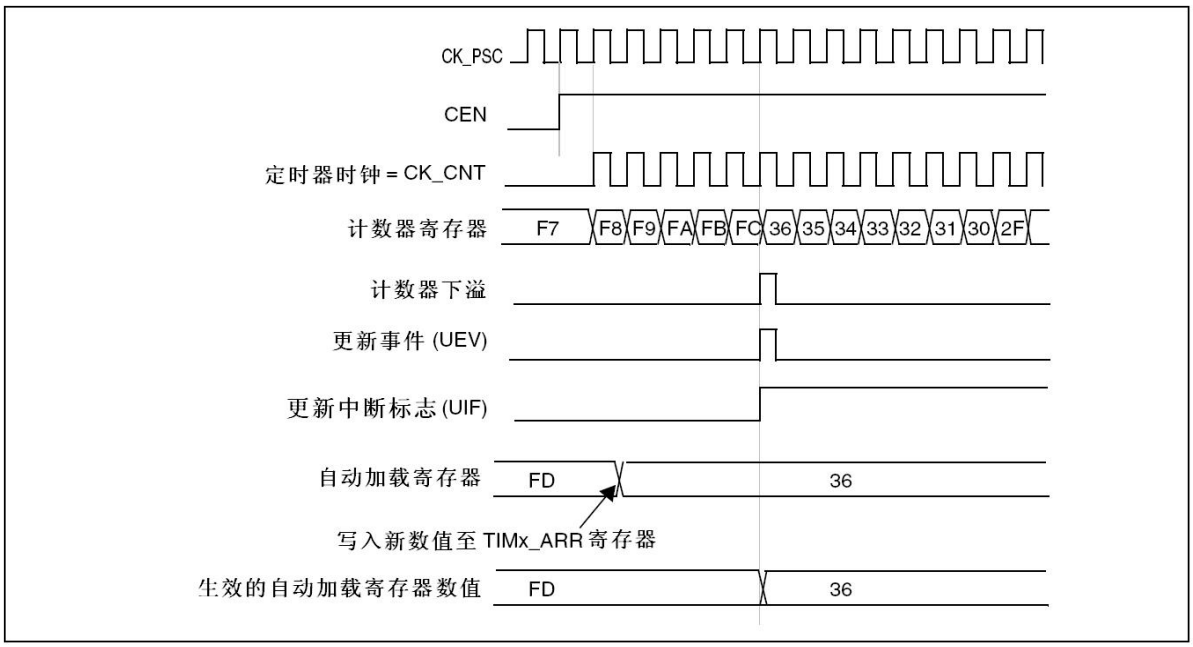


图 56 计数器时序图，ARPE=1 时的更新事件(计数器溢出)



12.3.3 重复计数器

13.3.1 节“时基单元”解释了计数器上溢/下溢时更新事件(UEV)是如何产生的，然而事实上它只能在重复计数达到 0 的时候产生。这个特性对产生 PWM 信号非常有用。

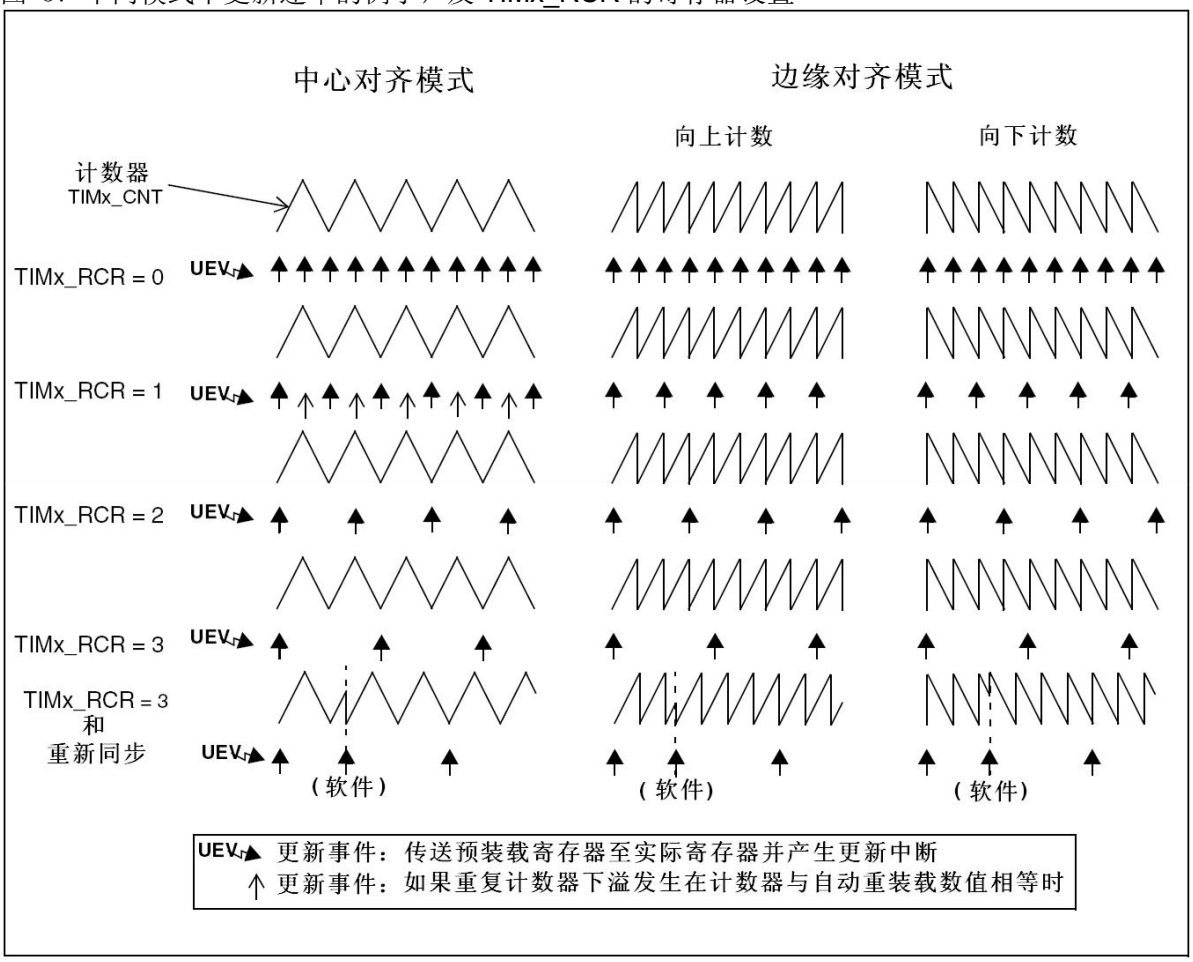
这意味着在每 N 次计数上溢或下溢时，数据从预装载寄存器传输到影子寄存器(TIMx_ARR 自动重载入寄存器，TIMx_PSC 预装载寄存器，还有在比较模式下的捕获/比较寄存器 TIMx_CCRx)，N 是 TIMx_RCR 重复计数寄存器中的值。

重复计数器在下述任一条件成立时递减：

- 向上计数模式下每次计数器溢出时，
- 向下计数模式下每次计数器下溢时，
- 中央对齐模式下每次上溢和每次下溢时。虽然这样限制了 PWM 的最大循环周期为 128，但它能够在每个 PWM 周期 2 次更新占空比。在中央对齐模式下，因为波形是对称的，如果每个 PWM 周期中仅刷新一次比较寄存器，则最大的分辨率为 $2xT_{ck}$。

重复计数器是自动加载的，重复速率是由 TIMx_RCR 寄存器的值定义(参看图 57)。当更新事件由软件产生(通过设置 TIMx_EGR 中的 UG 位)或者通过硬件的从模式控制器产生，则无论重复计数器的值是多少，立即发生更新事件，并且 TIMx_RCR 寄存器中的内容被重载入到重复计数器。

图 57 不同模式下更新速率的例子，及 TIMx_RCR 的寄存器设置



12.3.4 时钟选择

计数器时钟可由下列时钟源提供：

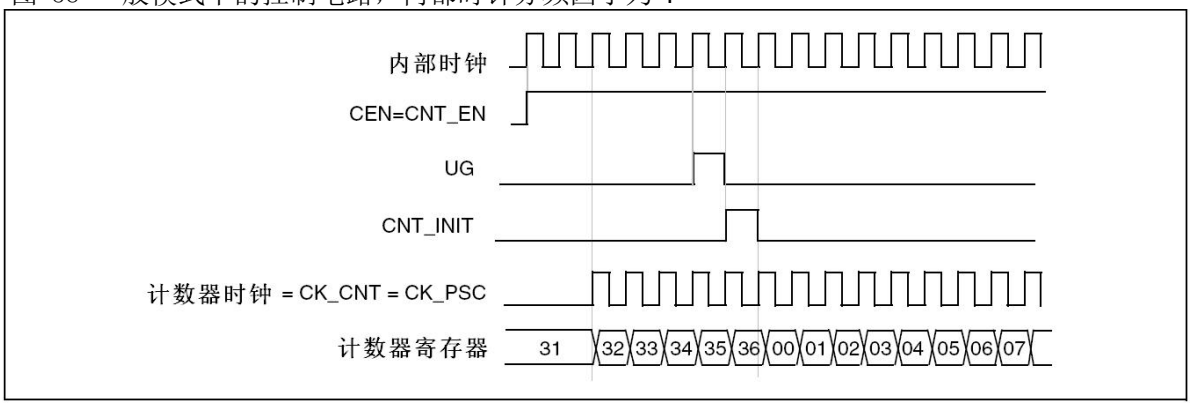
- 内部时钟(CK_INT)
- 外部时钟模式 1：外部输入引脚
- 外部时钟模式 2：外部触发输入 ETR
- 内部触发输入(ITRx)：使用一个定时器作为另一个定时器的预分频器。如可以配置一个定时器 Timer1 而作为另一个定时器 Timer2 的预分频器。详见下一章。

内部时钟源(CK_INT)

如果禁止了从模式控制器(SMS=000)，则 CEN、DIR(TIMx_CR1 寄存器)和 UG 位(TIMx_EGR 寄存器)是事实上的控制位，并且只能被软件修改(UG 位仍被自动清除)。只要 CEN 位被写成'1'，预分频器的时钟就由内部时钟 CK_INT 提供。

下图显示控制电路和向上计数器在一般模式下，不带预分频器时的操作。

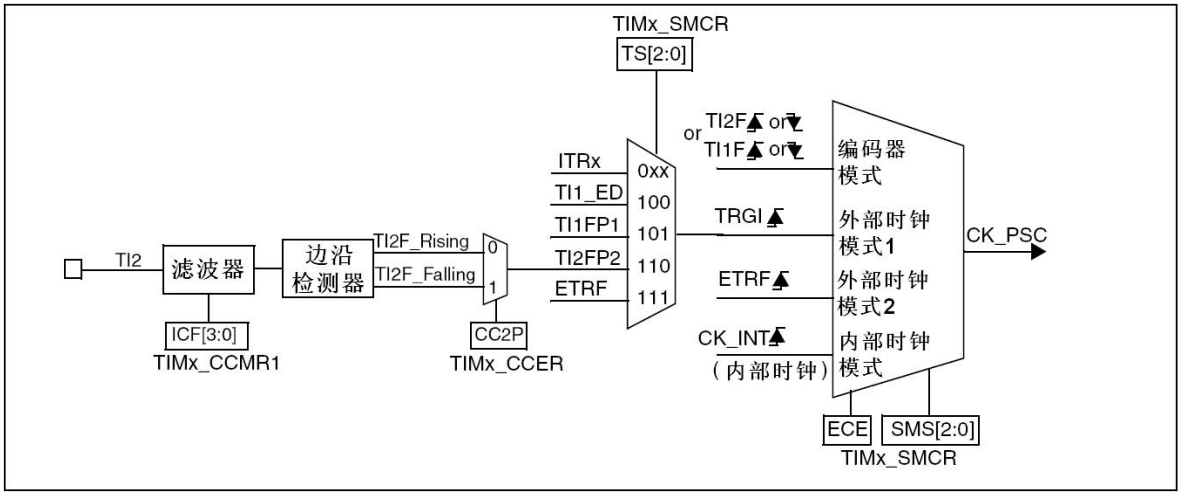
图 58 一般模式下的控制电路，内部时钟分频因子为 1



外部时钟源模式 1

当 TIMx_SMCR 寄存器的 SMS=111 时，此模式被选中。计数器可以在选定输入端的每个上升沿或下降沿计数。

图 59 TI2 外部时钟连接例子



例如，要配置向上计数器在 T12 输入端的上升沿计数，使用下列步骤：

配置 TIMx_CCMR1 寄存器 CC2S=01，

配置通道 2 检测 TI2 输入的上升沿

配置 TIMx_CCMR1 寄存器的 IC2F[3:0]，选择输入滤波器带宽(如果不需要滤波器，保持 IC2F=0000)

配置 TIMx_CCER 寄存器的 CC2P=0，选定上升沿极性

配置 TIMx_SMCR 寄存器的 SMS=111，选择定时器外部时钟模式 1

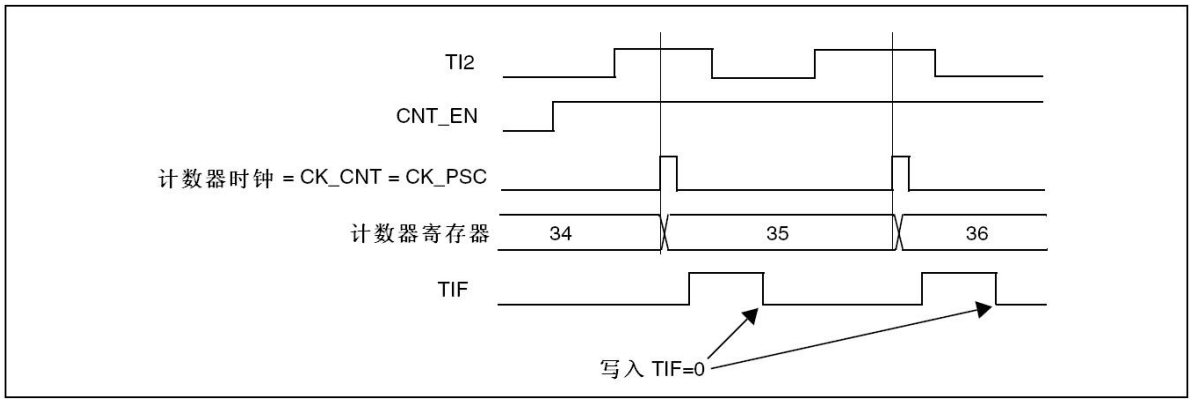
配置 TIMx_SMCR 寄存器中的 TS=110，选定 TI2 作为触发输入源设置 TIMx_CR1 寄存器的 CEN=1，启动计数器

注： *捕获预分频器不用作触发，所以不需要对它进行配置*

当上升沿出现在 TI2，计数器计数一次，且 TIF 标志被设置。

在 TI2 的上升沿和计数器实际时钟之间的延时，取决于在 TI2 输入端的重新同步电路。

图 60 外部时钟模式 1 下的控制电路



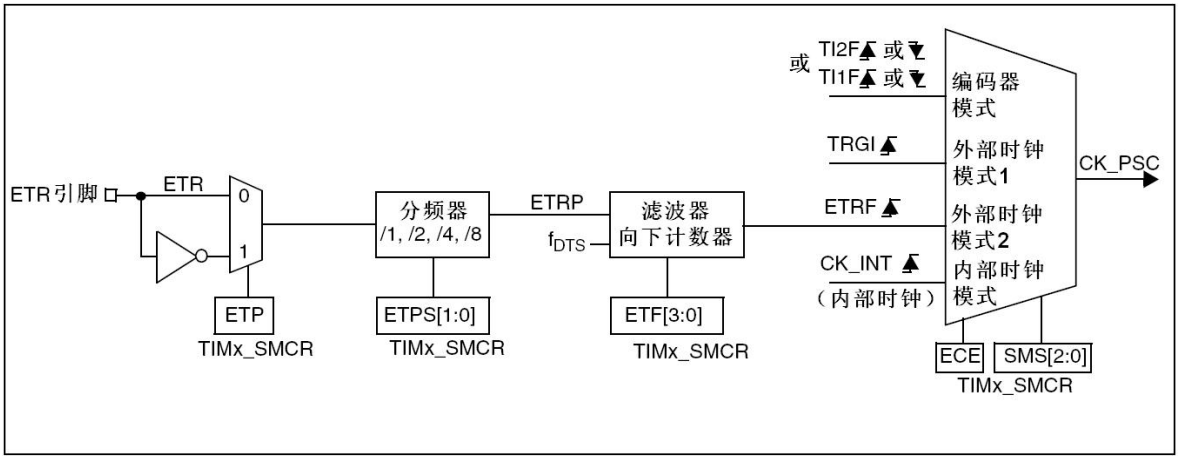
外部时钟源模式 2

选定此模式的方法为：令 TIMx_SMCR 寄存器中的 ECE=1

计数器能够在外部触发 ETR 的每一个上升沿或下降沿计数。

下图是外部触发输入的框图

图 61 外部触发输入框图



例如，要配置在 ETR 下每 2 个上升沿计数一次的向上计数器，使用下列步骤：

本例中不需要滤波器，置 TIMx_SMCR 寄存器中的 ETF[3:0]=0000

设置预分频器，置 TIMx_SMCR 寄存器中的 ETPS[1:0]=01

选择 ETR 的上升沿检测，置 TIMx_SMCR 寄存器中的 ETP=0

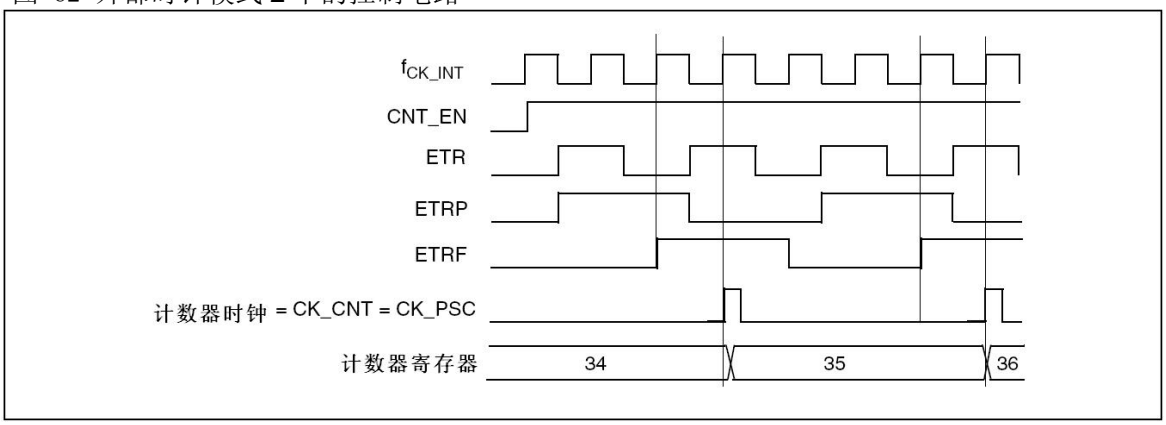
开启外部时钟模式 2，写 TIMx_SMCR 寄存器中的 ECE=1

启动计数器，写 TIMx_CR1 寄存器中的 CEN=1

计数器在每 2 个 ETR 上升沿计数一次。

在 ETR 的上升沿和计数器实际时钟之间的延时取决于在 ETRP 信号端的重新同步电路。

图 62 外部时钟模式 2 下的控制电路

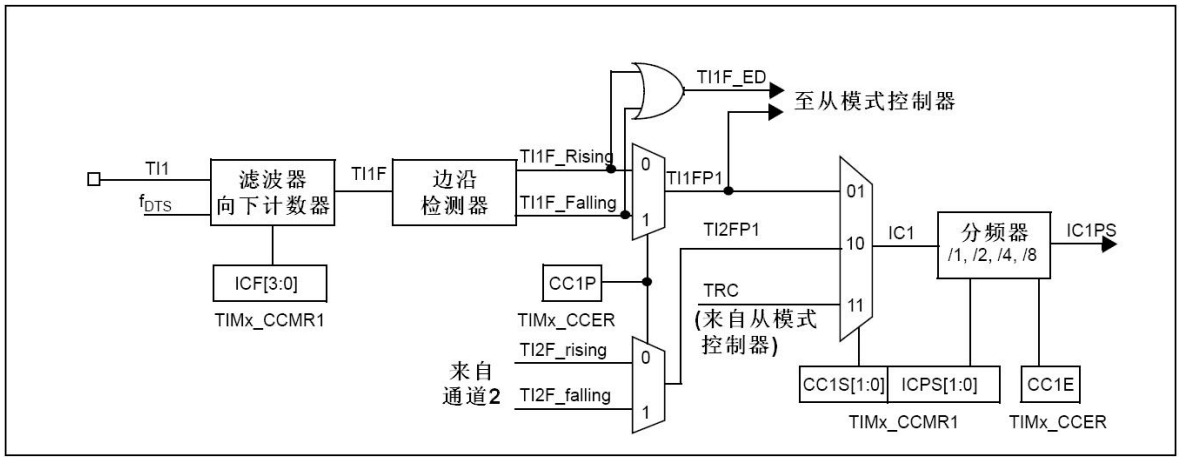


12.3.5 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器(包含影子寄存器)，包括捕获的输入部分(数字滤波、多路复用和预分频器)，和输出部分(比较器和输出控制)。 图 63 至图 66 是一个捕获/比较通道概览。

输入部分对相应的 TIx 输入信号采样，并产生一个滤波后的信号 TIxF。然后，一个带极性选择的边缘监测器产生一个信号(TIxFPx)，它可以作为从模式控制器的输入触发或者作为捕获控制。该信号通过预分频进入捕获寄存器(ICxPS)。

图 63 捕获/比较通道(如：通道 1 输入部分)



输出部分产生一个中间波形 OCxRef(高有效)作为基准，链的末端决定最终输出信号的极性。

图 64 捕获/比较通道 1 的主电路

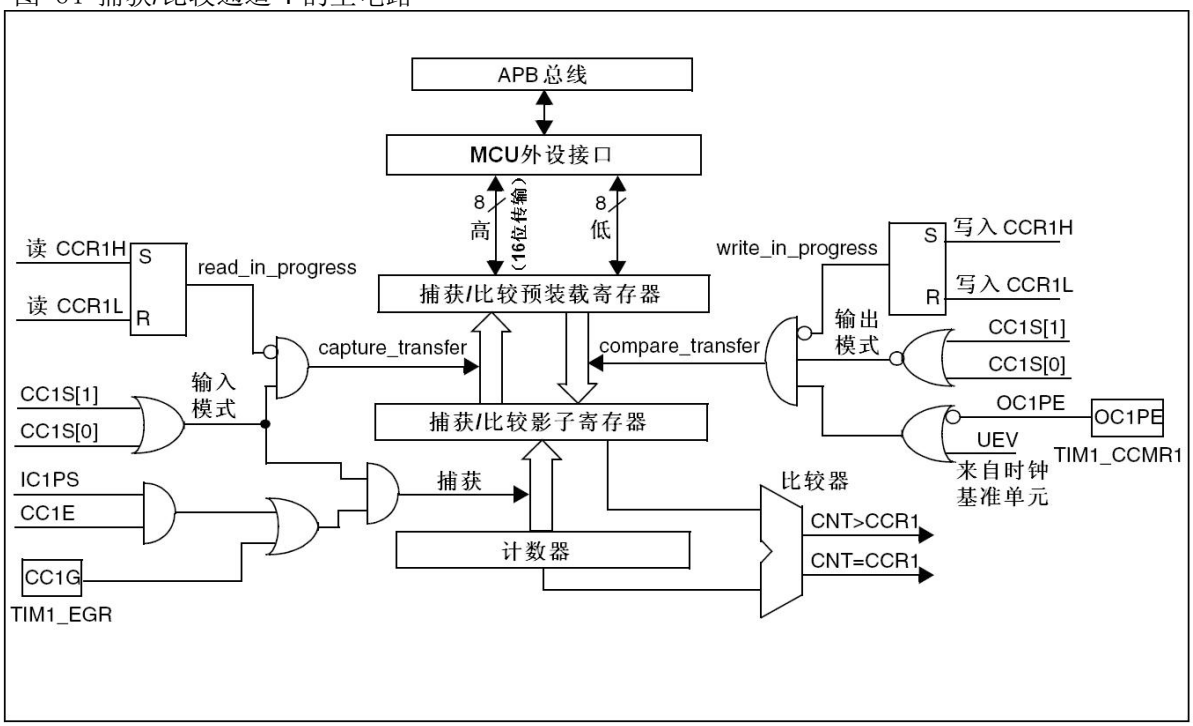


图 65 捕获/比较通道的输出部分(通道 1 至 3)

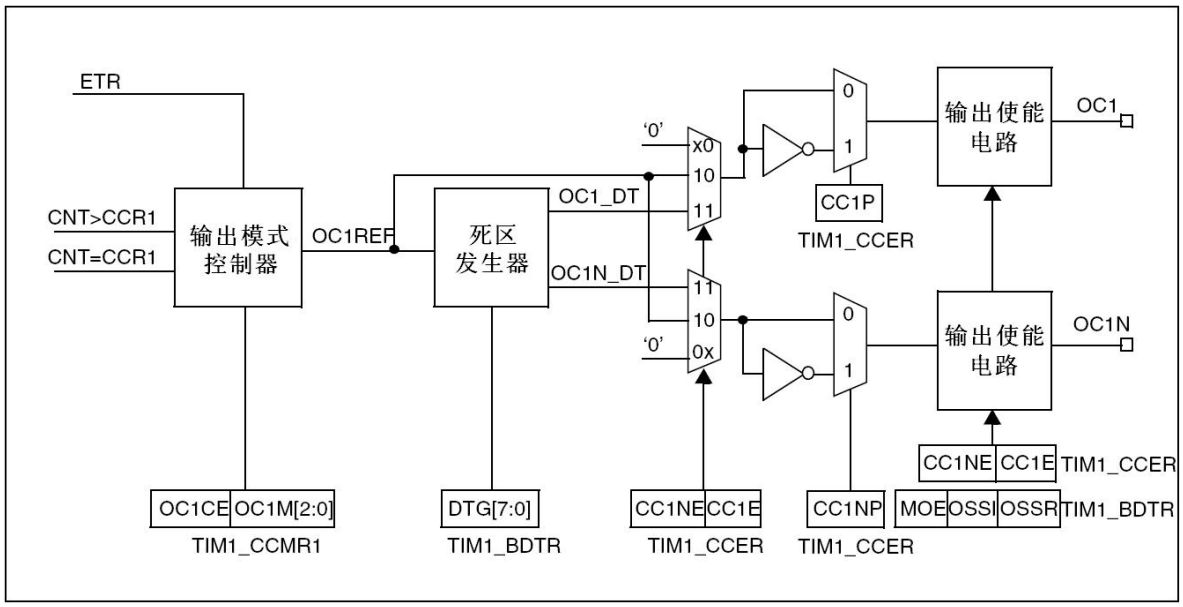
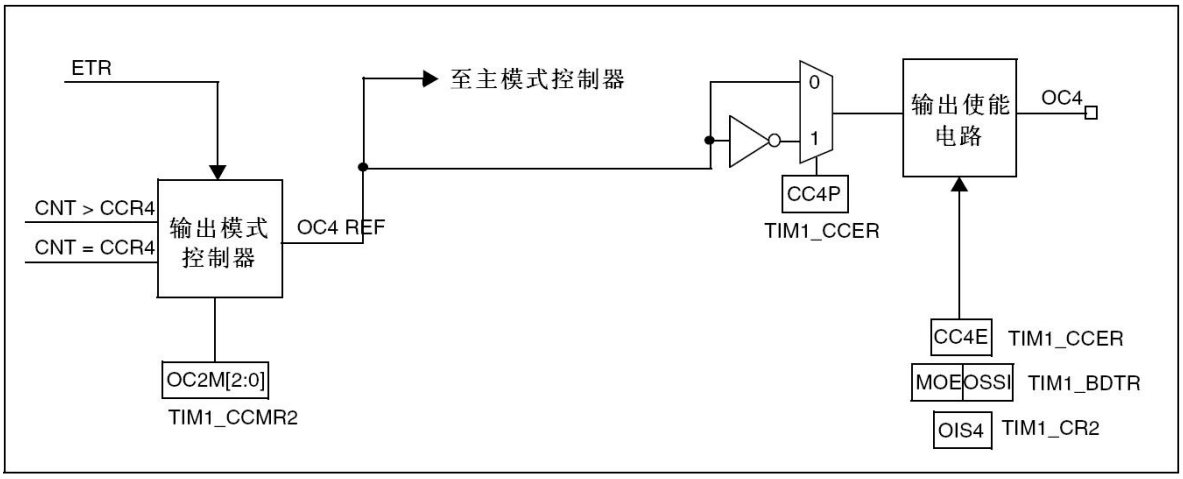


图 66 捕获/比较通道的输出部分(通道 4)



捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后影子寄存器的内容和计数器进行比较。

12.3.6 输入捕获模式

在输入捕获模式下，当检测到 ICx 信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器(TIMx_CCRx)中。当发生捕获事件时，相应的 CCxIF 标志(TIMx_SR 寄存器)被置 1，如果开放了中断或者 DMA 操作，则将产生中断或者 DMA 请求。如果发生捕获事件时 CCxIF 标志已经为高，那么重复捕获标志 CCxOF(TIMx_SR 寄存器)被置 1。写 CCxIF=0 可清除 CCxIF，或读取存储在 TIMx_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0 可清除 CCxOF。

以下例子说明如何在 TI1 输入的上升沿时捕获计数器的值到 TIMx_CCR1 寄存器中，步骤如下：

- 选择有效输入端：TIMx_CCR1 必须连接到 TI1 输入，所以写入 TIMx_CCR1 寄存器中的 CC1S=01，只要 CC1S 不为'00'，通道被配置为输入，并且 TIMx_CCR1 寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽(即输入为 TIx 时，输入滤波器控制位是 TIMx_CCMRx 寄存器中的 ICxF 位)。假设输入信号在最多 5 个内部时钟周期的时间内抖动，我们须配置滤波器的带宽长于 5 个时钟周期；因此我们可以(以 f_{DTS} 频率)连续采样 8 次，以确认在 TI1 上一次真实的边沿变换，即在 TIMx_CCMR1 寄存器中写入 IC1F=0011。
- 选择 TI1 通道的有效转换边沿，在 TIMx_CCER 寄存器中写入 CC1P=0(上升沿)。
- 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预 分频器被禁止(写 TIMx_CCMR1 寄存器的 IC1PS=00)。
- 设置 TIMx_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，通过设置 TIMx_DIER 寄存器中的 CC1IE 位允许相关中断请求，通过设置 TIMx_DIER 寄存器中的 CC1DE 位允许 DMA 请求。

当发生一个输入捕获时：

- 产生有效的电平转换时，计数器的值被传送到 TIMx_CCR1 寄存器。
- CC1IF 标志被设置(中断标志)。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除，CC1OF 也被置 1。
- 如设置了 CC1IE 位，则会产生一个中断。
- 如设置了 CC1DE 位，则还会产生一个 DMA 请求。 为了处理捕获溢出，建议在读出捕获溢出标志之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注：　设置 TIMx_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断和/或 DMA 请求。

12.3.7 PWM 输入模式

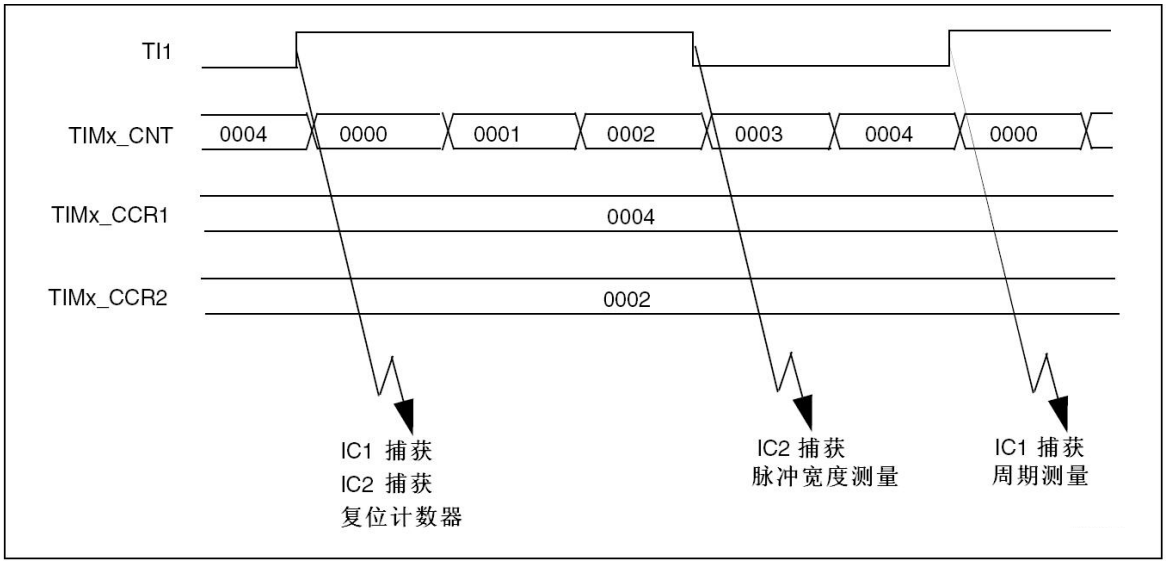
该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

- 两个 ICx 信号被映射至同一个 TIx 输入。
- 这 2 个 ICx 信号为边沿有效，但是极性相反。
- 其中一个 TIxFP 信号被作为触发输入信号，而从模式控制器被配置成复位模式。

例如，你需要测量输入到 TI1 上的 PWM 信号的长度(TIMx_CCR1 寄存器)和占空比(TIMx_CCR2 寄存器)，具体步骤如下(取决于 CK_INT 的频率和预分频器的值)

- 选择 TIMx_CCR1 的有效输入：置 TIMx_CCMR1 寄存器的 CC1S=01(选中 TI1)。
- 选择 TI1FP1 的有效极性(用来捕获数据到 TIMx_CCR1 中和清除计数器)：置 CC1P=0(上升沿有效)。
- 选择 TIMx_CCR2 的有效输入：置 TIMx_CCMR1 寄存器的 CC2S=10(选中 TI1)。
- 选择 TI1FP2 的有效极性(捕获数据到 TIMx_CCR2)：置 CC2P=1(下降沿有效)。
- 选择有效的触发输入信号：置 TIMx_SMCR 寄存器中的 TS=101(选择 TI1FP1)。
- 配置从模式控制器为复位模式：置 TIMx_SMCR 中的 SMS=100。
- 使能捕获：置 TIMx_CCER 寄存器中 CC1E=1 且 CC2E=1。

图 67 PWM 输入模式时序



因为只有 TI1FP1 和 TI2FP2 连到了从模式控制器，所以 PWM 输入模式只能使用 TIMx_CH1/TIMx_CH2 信号。

12.3.8 强置输出模式

在输出模式(TIMx_CCMRx 寄存器中 CCxS=00)下，输出比较信号(OCxREF 和相应的 OCx/OCxN)能够直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。

置 TIMx_CCMRx 寄存器中相应的 OCxM=101，即可强置输出比较信号(OCxREF/OCx)为有效状态。这样 OCxREF 被强置为高电平(OCxREF 始终为高电平有效)，同时 OCx 得到 CCxP 极性相反的信号。

例如：CCxP=0(OCx 高电平有效)，则 OCx 被强置为高电平。 置 TIMx_CCMRx 寄存器中的 OCxM=100，可强置 OCxREF 信号为低。

该模式下，在 TIMx_CCRx 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。因此仍然会产生相应的中断和 DMA 请求。这将会在下面的输出比较模式一节中介绍。

12.3.9 输出比较模式

此项功能是用来控制一个输出波形，或者指示一段给定的的时间已经到时。 当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作：

- 将输出比较模式(TIMx_CCMRx 寄存器中的 OCxM 位)和输出极性(TIMx_CCER 寄存器中的 CCxP 位)定义的值输出到对应的引脚上。在比较匹配时，输出引脚可以保持它的电平(OCxM=000)、被设置成有效电平(OCxM=001)、被设置成无效电平(OCxM=010)或进行翻转(OCxM=011)。
- 设置中断状态寄存器中的标志位(TIMx_SR 寄存器中的 CCxIF 位)。
- 若设置了相应的中断屏蔽(TIMx_DIER 寄存器中的 CCxIE 位)，则产生一个中断。
- 若设置了相应的使能位(TIMx_DIER 寄存器中的 CCxDE 位，TIMx_CR2 寄存器中的 CCDS 位选择 DMA 请求功能)，则产生一个 DMA 请求。

TIMx_CCMRx 中的 OCxPE 位选择 TIMx_CCRx 寄存器是否需要使用预装载寄存器。 在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。

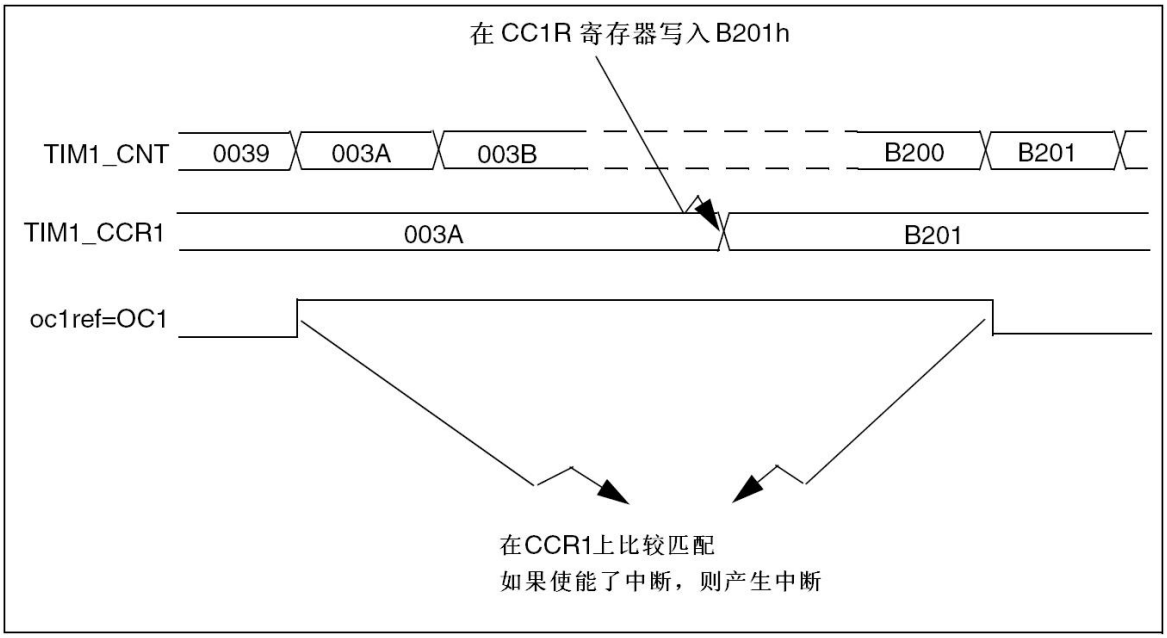
同步的精度可以达到计数器的一个计数周期。输出比较模式(在单脉冲模式下)也能用来输出一个单脉冲。

输出比较模式的配置步骤：

1. 选择计数器时钟(内部，外部，预分频器)。
2. 将相应的数据写入 TIMx_ARR 和 TIMx_CCRx 寄存器中。
3. 如果要产生一个中断请求，设置 CCxIE 位。
4. 选择输出模式，例如：
   - 要求计数器与 CCRx 匹配时翻转 OCx 的输出引脚，设置 OCxM=011
   - 置 OCxPE = 0 禁用预装载寄存器
   - 置 CCxP = 0 选择极性为高电平有效
   - 置 CCxE = 1 使能输出
5. 设置 TIMx_CR1 寄存器的 CEN 位启动计数器

TIMx_CCRx 寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄存器(OCxPE=’0’，否则 TIMx_CCRx 的影子寄存器只能在发生下一次更新事件时被更新)。下图给出了一个例子。

图 68 输出比较模式，翻转 OC1



12.3.10 PWM 模式

脉冲宽度调制模式可以产生一个由 TIMx_ARR 寄存器确定频率、由 TIMx_CCRx 寄存器确定占空比的信号。

在 TIMx_CCMRx 寄存器中的 OCxM 位写入'110'(PWM 模式 1)或'111'(PWM 模式 2)，能够独立地设置每个 OCx 输出通道产生一路 PWM。必须通过设置 TIMx_CCMRx 寄存器的 OCxPE 位使能相应的预装载寄存器，最后还要设置 TIMx_CR1 寄存器的 ARPE 位，(在向上计数或中心对称模式中)使能自动重装载的预装载寄存器。

仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，必须通过设置 TIMx_EGR 寄存器中的 UG 位来初始化所有的寄存器。 OCx 的极性可以通过软件在 TIMx_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或低电平有效。OCx 的输出使能通过(TIMx_CCER 和 TIMx_BDTR 寄存器中)CCxE、CCxNE、MOE、OSSI 和 OSSR 位的组合控制。详见 TIMx_CCER 寄存器的描述。

在 PWM 模式(模式 1 或模式 2)下，TIMx_CNT 和 TIMx_CCRx 始终在进行比较，(依据计数器的计数 方向)以确定是否符合 TIMx_CCRx≤TIMx_CNT 或者 TIMx_CNT≤TIMx_CCRx。

根据 TIMx_CR1 寄存器中 CMS 位的状态，定时器能够产生边沿对齐的 PWM 信号或中央对齐的 PWM 信号。

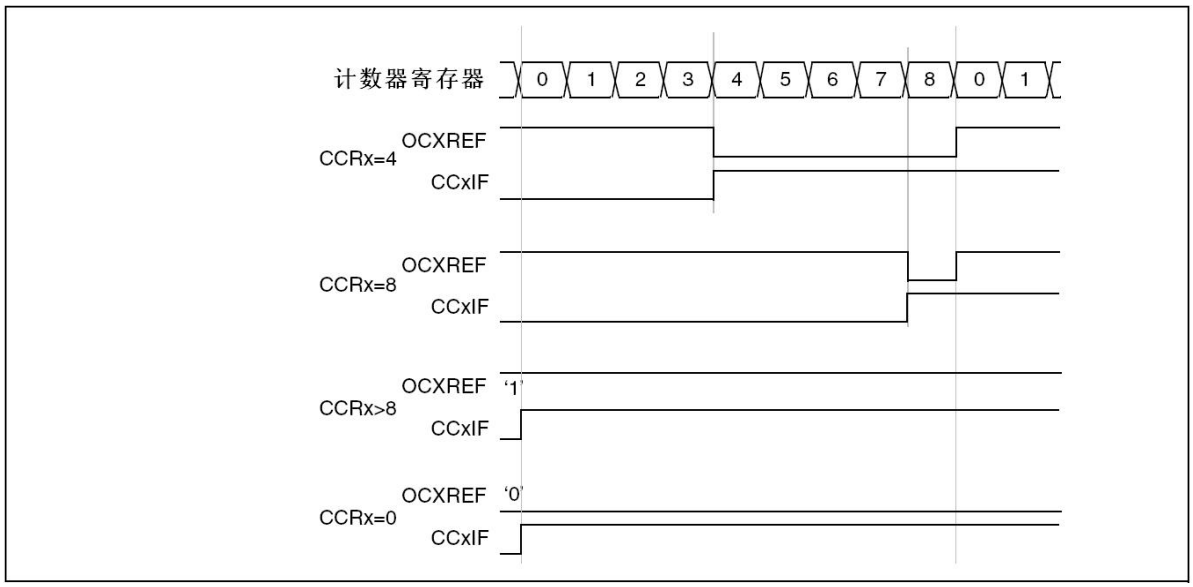
PWM 边沿对齐模式

- 向上计数配置

  当 TIMx_CR1 寄存器中的 DIR 位为低的时候执行向上计数。参看 13.3.2 节。

  下面是一个 PWM 模式 1 的例子。当 TIMx_CNT<TIMx_CCRx 时，PWM 参考信号 OCxREF 为 高，否则为低。如果 TIMx_CCRx 中的比较值大于自动重装载值(TIMx_ARR)，则 OCxREF 保持为'1'。如果比较值为 0，则 OCxREF 保持为'0'。下图为 TIMx_ARR=8 时边沿对齐的 PWM 波形实例。

图 69 边沿对齐的 PWM 波形(ARR=8)



- 向下计数的配置

  当 TIMx_CR1 寄存器的 DIR 位为高时执行向下计数。参看 13.3.2 节。

  在 PWM 模式 1，当 TIMx_CNT>TIMx_CCRx 时参考信号 OCxREF 为低，否则为高。如果 TIMx_CCRx 中的比较值大于 TIMx_ARR 中的自动重装载值，则 OCxREF 保持为'1'。该模式下不能产生 0%的 PWM 波形。

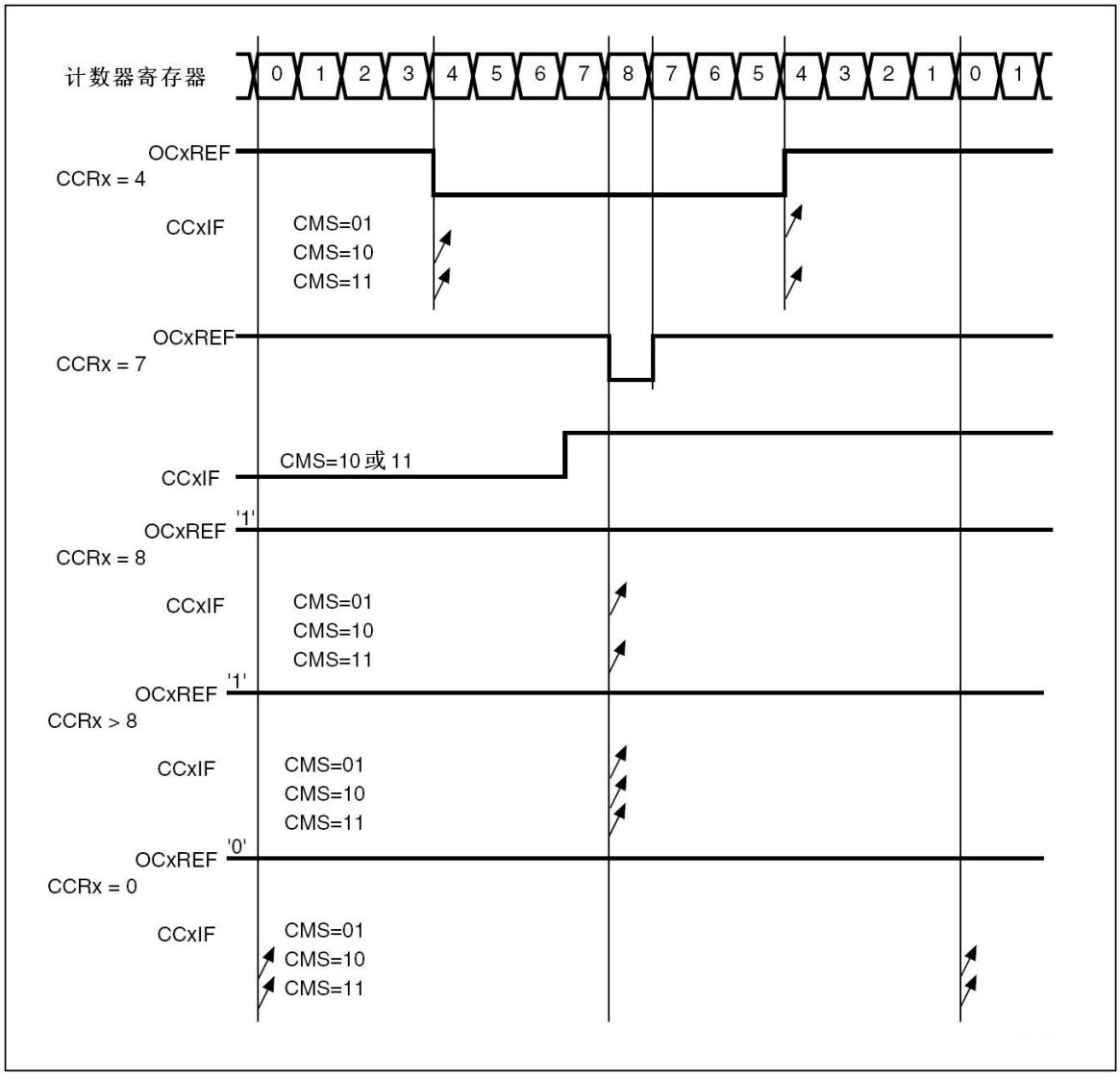
PWM 中央对齐模式

当 TIMx_CR1 寄存器中的 CMS 位不为'00'时为中央对齐模式(所有其他的配置对 OCxREF/OCx 信号都有相同的作用)。根据不同的 CMS 位设置，比较标志可以在计数器向上计数时被置 1、在计数器向下计数时被置 1、或在计数器向上和向下计数时被置 1。TIMx_CR1 寄存器中的计数方向位(DIR)由硬件更新，不要用软件修改它。参看 13.3.2 节的中央对齐模式。

下图给出了一些中央对齐的 PWM 波形的例子

- TIMx_ARR=8
- PWM 模式 1
- TIMx_CR1 寄存器的 CMS=01，在中央对齐模式 1 下，当计数器向下计数时设置比较标志。

图 70 中央对齐的 PWM 波形(APR=8)



使用中央对齐模式的提示

- 进入中央对齐模式时，使用当前的向上/向下计数配置；这就意味着计数器向上还是向下计数取决于 TIMx_CR1 寄存器中 DIR 位的当前值。此外，软件不能同时修改 DIR 和 CMS 位。
- 不推荐当运行在中央对齐模式时改写计数器，因为这会产生不可预知的结果。特别地：
  - 如果写入计数器的值大于自动重加载的值(TIMx_CNT>TIMx_ARR)，则方向不会被更新。例如，如果计数器正在向上计数，它就会继续向上计数。
  - 如果将 0 或者 TIMx_ARR 的值写入计数器，方向被更新，但不产生更新事件 UEV。

- 使用中央对齐模式最保险的方法，就是在启动计数器之前产生一个软件更新(设置 TIMx_EGR 位中的 UG 位)，并且不要在计数进行过程中修改计数器的值。

12.3.11 互补输出和死区插入

高级控制定时器(TIM1)能够输出两路互补信号，并且能够管理输出的瞬时关断和接通。

这段时间通常被称为死区，用户应该根据连接的输出器件和它们的特性(电平转换的延时、电源开关的延时等)来调整死区时间。

配置 TIMx_CCER 寄存器中的 CCxP 和 CCxNP 位，可以为每一个输出独立地选择极性(主输出 OCx 或互补输出 OCxN)。

互补信号 OCx 和 OCxN 通过下列控制位的组合进行控制：TIMx_CCER 寄存器的 CCxE 和 CCxNE 位，TIMx_BDTR 和 TIMx_CR2 寄存器中的 MOE、OISx、OISxN、OSSI 和 OSSR 位，详见表 53 带刹车功能的互补输出通道 OCx 和 OCxN 的控制位。特别的是，在转换到 IDLE 状态时(MOE 下降到 0)死区被激活。

同时设置 CCxE 和 CCxNE 位将插入死区，如果存在刹车电路，则还要设置 MOE 位。每一个通道都有一个 10 位的死区发生器。参考信号 OCxREF 可以产生 2 路输出 OCx 和 OCxN。

如果 OCx 和OCxN 为高有效：

- OCx 输出信号与参考信号相同，只是它的上升沿相对于参考信号的上升沿有一个延迟。
- OCxN 输出信号与参考信号相反，只是它的上升沿相对于参考信号的下降沿有一个延迟。 如果延迟大于当前有效的输出宽度(OCx 或者 OCxN)，则不会产生相应的脉冲。

下列几张图显示了死区发生器的输出信号和当前参考信号 OCxREF 之间的关系。(假设 CCxP=0、CCxNP=0、MOE=1、CCxE=1 并且 CCxNE=1)

图 71 带死区插入的互补输出

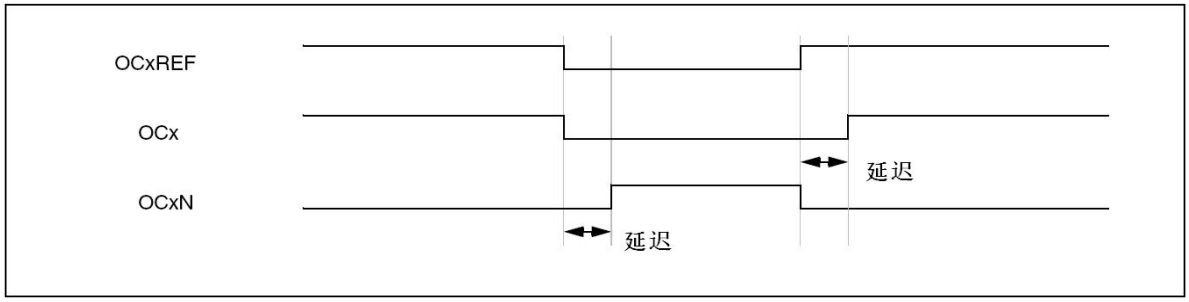


图 72 死区波形延迟大于负脉冲

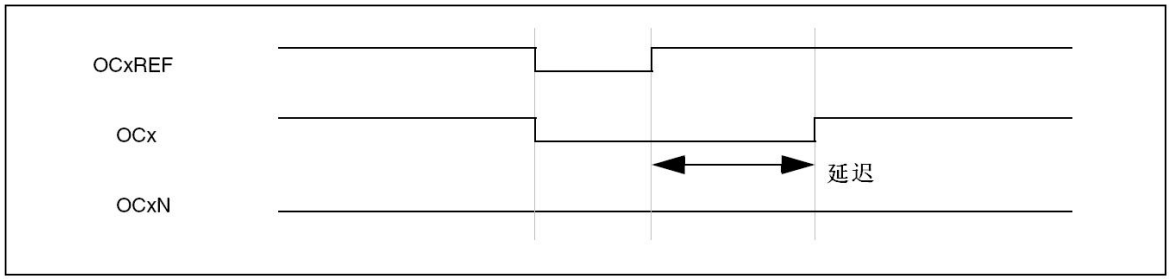
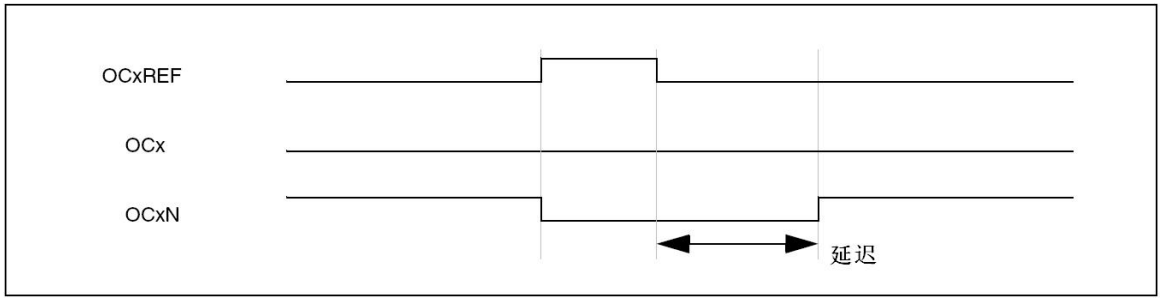


图 73 死区波形延迟大于正脉冲



每一个通道的死区延时都是相同的，是由TIMx_BDTR寄存器中的DTG位编程配置。详见 13.4.18 节 TIM1 刹车和死区寄存器(TIMx_BDTR)中的延时计算。

重定向 OCxREF 到 OCx 或 OCxN

在输出模式下(强置、输出比较或 PWM)，通过配置 TIMx_CCER 寄存器的 CCxE 和 CCxNE 位，OCxREF 可以被重定向到 OCx 或者 OCxN 的输出。

这个功能可以在互补输出处于无效电平时，在某个输出上送出一个特殊的波形(例如 PWM 或者静态有效电平)。另一个作用是，让两个输出同时处于无效电平，或处于有效电平和带死区的互补输出。

注：当只使能 OCxN(CCxE=0,CCxNE=1)时，它不会反相，当 OCxREF 有效时立即变高。例如，如果 CCxNP=0，则 OCxN=OCxREF。另一方面，当 OCx 和 OCxN 都被使能时(CCxE=CCxNE=1)，当 OCxREF 为高时 OCx 有效；而 OCxN 相反，当 OCxREF 低时 OCxN 变为有效。

12.3.12 使用刹车功能

当使用刹车功能时，依据相应的控制位(TIMx_BDTR 寄存器中的 MOE、OSSI 和 OSSR 位，TIMx_CR2 寄存器中的 OISx 和 OISxN 位)，输出使能信号和无效电平都会被修改。但无论何时， OCx 和 OCxN 输出不能在同一时间同时处于有效电平上。详见表 53 带刹车功能的互补输出通道OCx 和 OCxN 的控制位。

刹车源既可以是刹车输入引脚又可以是一个时钟失败事件。时钟失败事件由复位时钟控制器中的时钟安全系统产生，详见 8.2.7 节时钟安全系统(CSS)。 系统复位后，刹车电路被禁止，MOE 位为低。设置 TIMx_BDTR 寄存器中的 BKE 位可以使能刹车
功能，刹车输入信号的极性可以通过配置同一个寄存器中的 BKP 位选择。BKE 和 BKP 可以同时被修改。当写入 BKE 和 BKP 位时，在真正写入之前会有 1 个 APB 时钟周期的延迟，因此需要等待一个 APB 时钟周期之后，才能正确地读回写入的位。

因为 MOE 下降沿可以是异步的，在实际信号(作用在输出端)和同步控制位(在 TIMx_BDTR 寄存器 中)之间设置了一个再同步电路。这个再同步电路会在异步信号和同步信号之间产生延迟。特别的，如果当它为低时写 MOE=1，则读出它之前必须先插入一个延时(空指令)才能读到正确的值。这是因为写入的是异步信号而读的是同步信号。

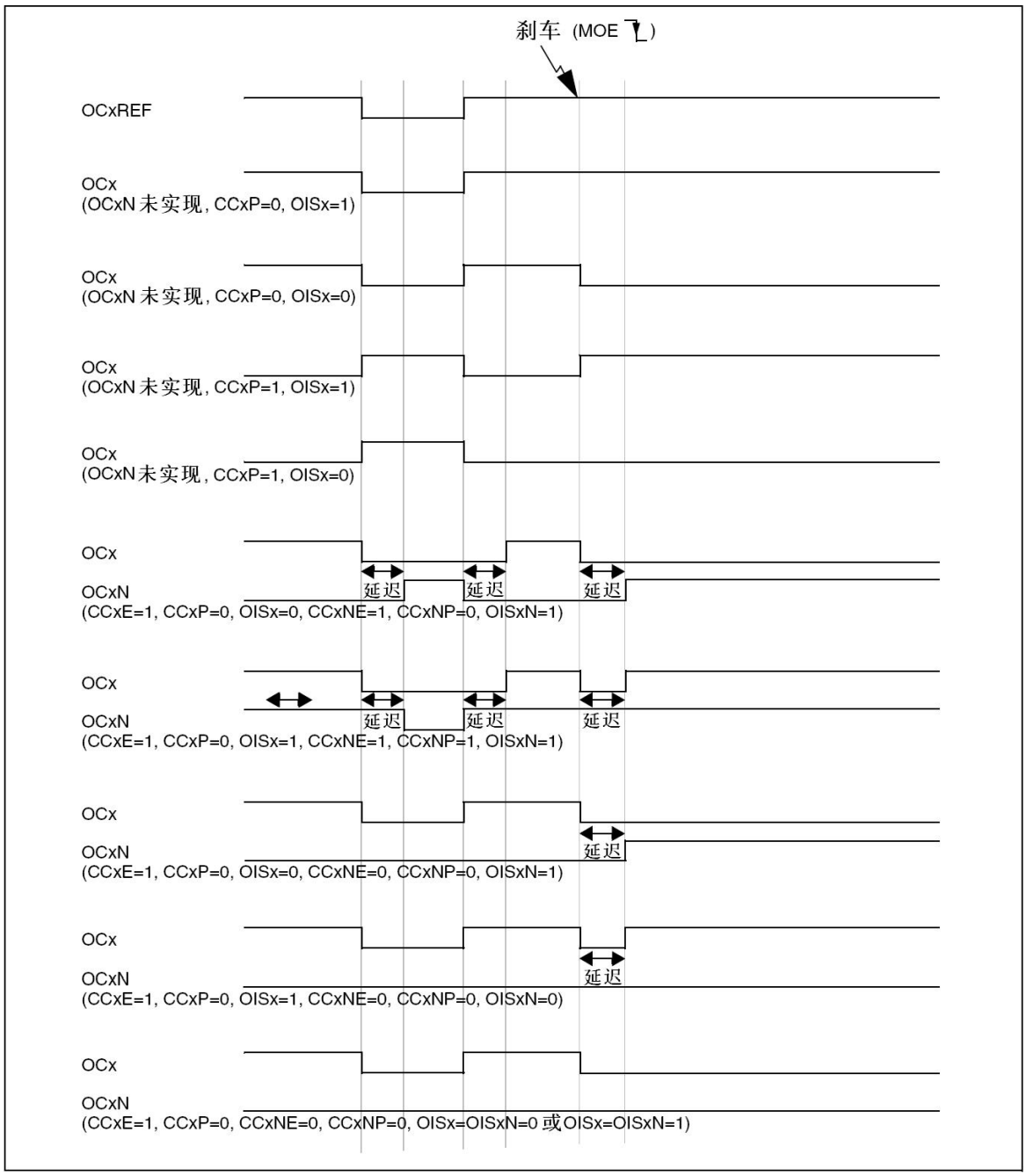
当发生刹车时(在刹车输入端出现选定的电平)，有下述动作：

- MOE 位被异步地清除，将输出置于无效状态、空闲状态或者复位状态(由 OSSI 位选择)。这个特性在 MCU 的振荡器关闭时依然有效。
- 一旦 MOE=0，每一个输出通道输出由 TIMx_CR2 寄存器中的 OISx 位设定的电平。如果 OSSI=0，则定时器释放使能输出，否则使能输出始终为高。
- 当使用互补输出时：
  - 输出首先被置于复位状态即无效的状态(取决于极性)。这是异步操作，即使定时器没有时钟时，此功能也有效。
  - 如果定时器的时钟依然存在，死区生成器将会重新生效，在死区之后根据 OISx 和 OISxN 位指示的电平驱动输出端口。即使在这种情况下，OCx 和 OCxN 也不能被同时驱动到有效 的电平。注，因为重新同步 MOE，死区时间比通常情况下长一些(大约 2 个 ck_tim 的时钟周期)。
  - 如果 OSSI=0，定时器释放使能输出，否则保持使能输出；或一旦 CCxE 与 CCxNE 之一变 高时，使能输出变为高。
- 如果设置了 TIMx_DIER 寄存器中的 BIE 位，当刹车状态标志(TIMx_SR 寄存器中的 BIF 位)为'1'时，则产生一个中断。如果设置了 TIMx_DIER 寄存器中的 BDE 位，则产生一个 DMA 请求。
- 如果设置了 TIMx_BDTR 寄存器中的 AOE 位，在下一个更新事件 UEV 时 MOE 位被自动置位； 例如，这可以用来进行整形。否则，MOE 始终保持低直到被再次置'1'；此时，这个特性可以被用在安全方面，你可以把刹车输入连到电源驱动的报警输出、热敏传感器或者其他安全器件上。

注：　刹车输入为电平有效。所以，当刹车输入有效时，不能同时(自动地或者通过软件)设置 MOE。同时，状态标志 BIF 不能被清除。

刹车由 BRK 输入产生，它的有效极性是可编程的，且由 TIMx_BDTR 寄存器中的 BKE 位开启。除了刹车输入和输出管理，刹车电路中还实现了写保护以保证应用程序的安全。它允许用户冻结几个配置参数(死区长度，OCx/OCxN 极性和被禁止的状态，OCxM 配置，刹车使能和极性)。用户可以通过 TIMx_BDTR 寄存器中的 LOCK 位，从三级保护中选择一种，参看 13.3.18 节 TIM1 刹车和死区寄存器(TIMx_BDTR)。在 MCU 复位后 LOCK 位只能被修改一次。下图显示响应刹车的输出实例。

图 74 响应刹车的输出



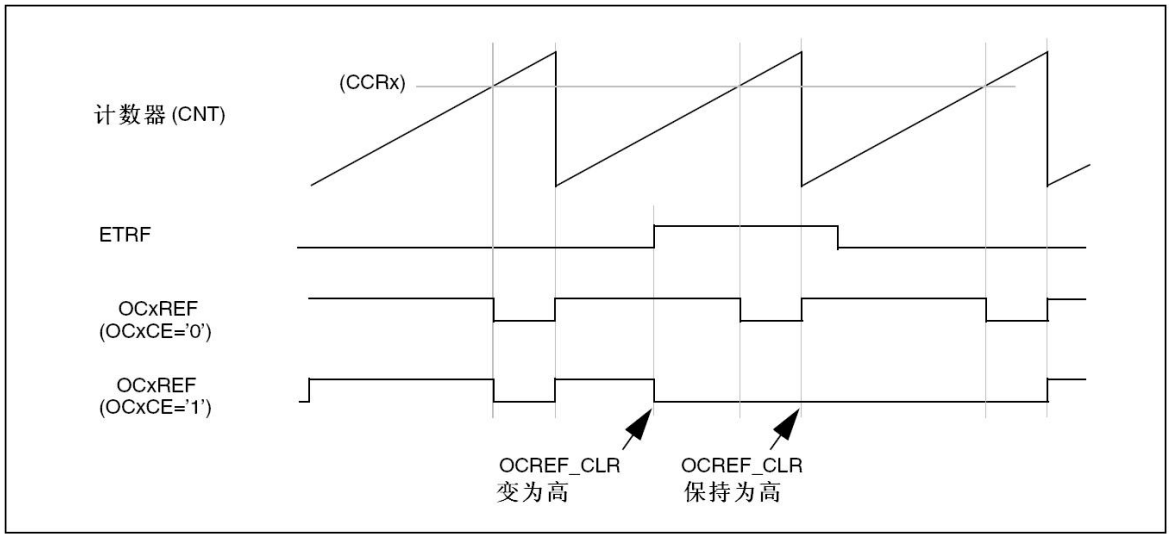
12.3.13 在外部事件时清除 OCxREF 信号

对于一个给定的通道，设置 TIMx_CCMRx 寄存器中对应的 OCxCE 位为'1'，能够用 ETRF 输入端 的高电平把 OCxREF 信号拉低，OCxREF 信号将保持为低直到发生下一次的更新事件 UEV。该功能只能用于输出比较和 PWM 模式，而不能用于强置模式。 例如，OCxREF 信号可以联到一个比较器的输出，用于控制电流。这时，ETR 必须配置如下：

1. 外部触发预分频器必须处于关闭：TIMx_SMCR 寄存器中的 ETPS[1:0]=00。
2. 必须禁止外部时钟模式 2：TIMx_SMCR 寄存器中的 ECE=0。
3. 外部触发极性(ETP)和外部触发滤波器(ETF)可以根据需要配置。

下图显示了当 ETRF 输入变为高时，对应不同 OCxCE 的值，OCxREF 信号的动作。在这个例子中，定时器 TIMx 被置于 PWM 模式。

图 75 清除 TIMx 的 OCxREF



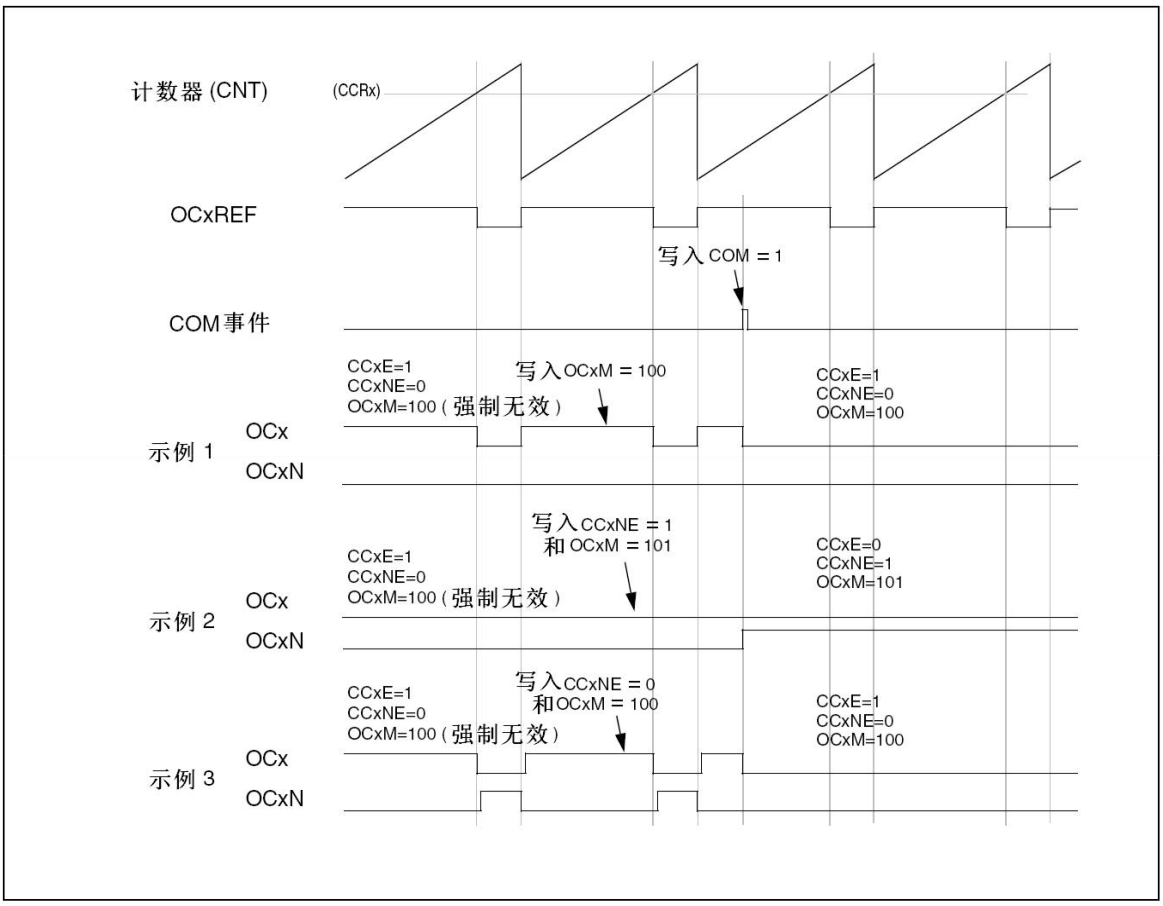
12.3.14 产生六步 PWM 输出

当在一个通道上需要互补输出时，预装载位有 OCxM、CCxE 和 CCxNE。在发生 COM 换相事件时，这些预装载位被传送到影子寄存器位。这样你就可以预先设置好下一步骤配置，并在同一个时刻同时修更改所有通道的配置。COM 可以通过设置 TIMx_EGR 寄存器的 COM 位由软件产生，或在 TRGI 上升沿由硬件产生。

当发生 COM 事件时会设置一个标志位(TIMx_SR 寄存器中的 COMIF 位)，这时如果已设置了 TIMx_DIER 寄存器的 COMIE 位，则产生一个中断；如果已设置了 TIMx_DIER 寄存器的 COMDE 位，则产生一个 DMA 请求。

下图显示当发生 COM 事件时，三种不同配置下 OCx 和 OCxN 输出。

图 76 产生六步 PWM，使用 COM 的例子(OSSR=1)



12.3.15 单脉冲模式

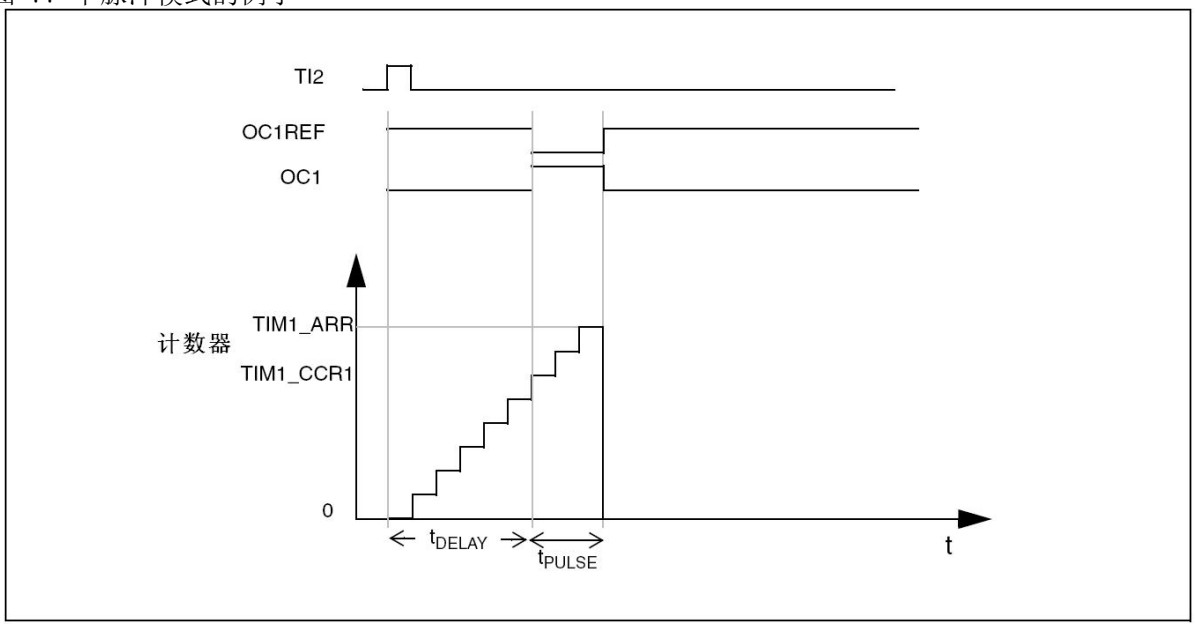
单脉冲模式(OPM)是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后产生一个脉宽可程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TIMx_CR1 寄存器中的 OPM 位将选择单脉冲模式，这样可以让计数器自动地在产生下一个更新事件 UEV 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前(当定时器正在等待触发)，必须如下配置：

- 向上计数方式：计数器 CNT < CCRx ≤ ARR (特别地, 0 < CCRx)，
- 向下计数方式：计数器 CNT > CCRx。

图 77 单脉冲模式的例子



例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 t_{DELAY} 之后，在 OC1 上产生一个长度为 t_{PULSE} 的正脉冲。

假定 TI2FP2 作为触发 1:

- 置 TIMx_CCMR1 寄存器中的 CC2S=01，把 TI2FP2 映像到 TI2。
- 置 TIMx_CCER 寄存器中的 CC2P=0，使 TI2FP2 能够检测上升沿。
- 置 TIMx_SMCR 寄存器中的 TS=110，TI2FP2 作为从模式控制器的触发(TRGI)。
- 置 TIMx_SMCR 寄存器中的 SMS=110(触发模式)，TI2FP2 被用来启动计数器。

OPM 的波形由写入比较寄存器的数值决定(要考虑时钟频率和计数器预分频器)

- t_{DELAY} 由 TIMx_CCR1 寄存器中的值定义。
- t_{PULSE} 由自动装载值和比较值之间的差值定义(TIMx_ARR - TIMx_CCR1)。
- 假定当发生比较匹配时要产生从 0 到 1 的波形，当计数器达到预装载值时要产生一个从 1 到 0 的波形；首先要置 TIMx_CCMR1 寄存器的 OC1M=111，进入 PWM 模式 2；根据需要有选择 地使能预装载寄存器：置 TIMx_CCMR1 中的 OC1PE=1 和 TIMx_CR1 寄存器中的 ARPE；然 后在 TIMx_CCR1 寄存器中填写比较值，在 TIMx_ARR 寄存器中填写自动装载值，设置 UG 位 来产生一个更新事件，然后等待在 TI2 上的一个外部触发事件。本例中，CC1P=0。

在这个例子中，TIMx_CR1 寄存器中的 DIR 和 CMS 位应该置低。

因为只需要一个脉冲，所以必须设置 TIMx_CR1 寄存器中的 OPM=1，在下一个更新事件(当计数器从自动装载值翻转到 0)时停止计数。

特殊情况：OCx 快速使能：

在单脉冲模式下，在 TIx 输入脚的边沿检测逻辑设置 CEN 位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期，因此它限制了可得到的最小延时 t_{DELAY}。 如果要以最小延时输出波形，可以设置 TIMx_CCMRx 寄存器中的 OCxFE 位；此时 OCxREF(和OCx)直接响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。OCxFE 只在通道配置为 PWM1 和 PWM2 模式时起作用。

12.3.16 编码器接口模式

选择编码器接口模式的方法是：如果计数器只在 TI2 的边沿计数，则置 TIMx_SMCR 寄存器中的SMS=001；如果只在 TI1 边沿计数，则置 SMS=010；如果计数器同时在 TI1 和 TI2 边沿计数，则置 SMS=011。

通过设置 TIMx_CCER 寄存器中的 CC1P 和 CC2P 位，可以选择 TI1 和 TI2 极性；如果需要，还可以 对输入滤波器编程。

两个输入 TI1 和 TI2 被用来作为增量编码器的接口。参看表 51，假定计数器已经启动(TIMx_CR1 寄存器中的 CEN=1)，则计数器由每次在 TI1FP1 或 TI2FP2 上的有效跳变驱动。TI1FP1 和 TI2FP2 是 TI1 和 TI2 在通过输入滤波器和极性控制后的信号；如果没有滤波和变相，则 TI1FP1=TI1，TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号 的跳变顺序，计数器向上或向下计数，同时硬件对 TIMx_CR1 寄存器的 DIR 位进行相应的设置。不管计数器是依靠 TI1 计数、依靠 TI2 计数或者同时依靠 TI1 和 TI2 计数，在任一输入端(TI1 或者TI2)的跳变都会重新计算 DIR 位。

编码器接口模式基本上相当于使用了一个带有方向选择的外部时钟。这意味着计数器只在 0 到TIMx_ARR 寄存器的自动装载值之间连续计数(根据方向，或是 0 到 ARR 计数，或是 ARR 到 0 计数)。所以在开始计数之前必须配置 TIMx_ARR；同样，捕获器、比较器、预分频器、重复计数器、触发输出特性等仍工作如常。编码器模式和外部时钟模式 2 不兼容，因此不能同时操作。

在这个模式下，计数器依照增量编码器的速度和方向被自动的修改，因此计数器的内容始终指示着编码器的位置。计数方向与相连的传感器旋转的方向对应。下表列出了所有可能的组合，假设 TI1 和 TI2 不同时变换。

表 51 计数方向与编码器信号的关系

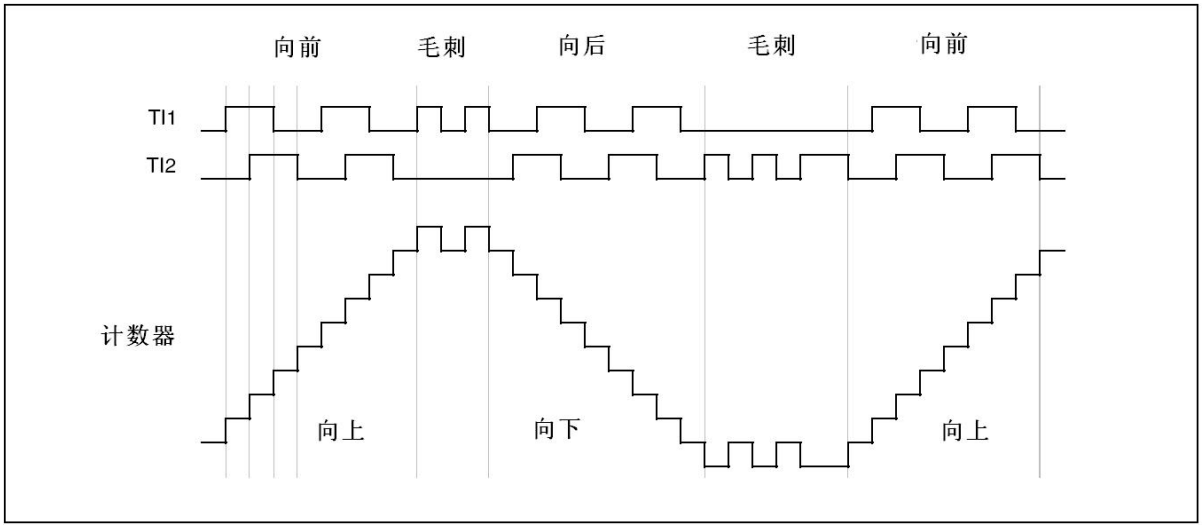
有效边沿	相对信号的电平 (TI1FP1 对应 TI2, TI2FP2 对	TI1FP1 信号		TI2FP2 信号	
		上升	下降	上升	下降
仅在 TI1 计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在 TI2 计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在 TI1 和 TI2 上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

一个外部的增量编码器可以直接与 MCU 连接而不需要外部接口逻辑。但是，一般会使用比较器将编码器的差动输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信号表示机械零点，可以把它连接到一个外部中断输入并触发一个计数器复位。

下图是一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；抖动可能会在传感器的位置靠近一个转换点时产生。在这个例子中，我们假定配置如下：

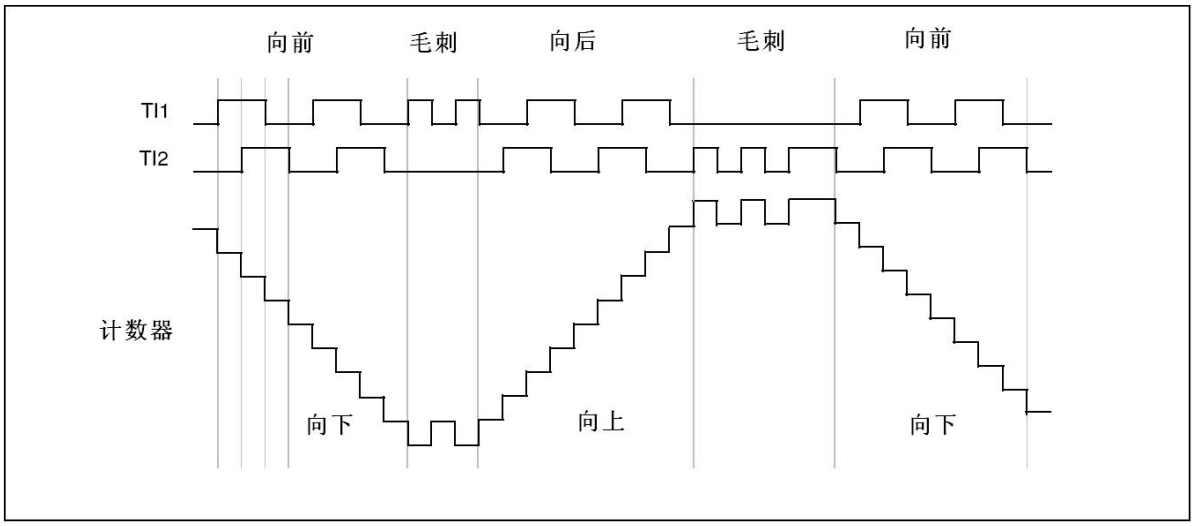
- CC1S=’01’ (TIMx_CCMR1 寄存器，IC1FP1 映射到 TI1)
- CC2S=’01’ (TIMx_CCMR2 寄存器，IC2FP2 映射到 TI2)
- CC1P=’0’(TIMx_CCER 寄存器，IC1FP1 不反相，IC1FP1=TI1)
- CC2P=’0’(TIMx_CCER 寄存器，IC2FP2 不反相，IC2FP2=TI2)
- SMS=’011’(TIMx_SMCR 寄存器，所有的输入均在上升沿和下降沿有效).
- CEN=’1’(TIMx_CR1 寄存器，计数器使能)

图 78 编码器模式下的计数器操作实例



下图为当 IC1FP1 极性反相时计数器的操作实例(CC1P='1'，其他配置与上例相同)

图 79 IC1FP1 反相的编码器接口模式实例



当定时器配置成编码器接口模式时，提供传感器当前位置的信息。使用第二个配置在捕获模式的定时器，可以测量两个编码器事件的间隔，获得动态的信息(速度，加速度，减速度)。指示机械零点的编码器输出可被用做此目的。根据两个事件间的间隔，可以按照固定的时间读出计数器。如果可能的话，你可以把计数器的值锁存到第三个输入捕获寄存器(捕获信号必须是周期的并且可以由另一个定时器产生)；也可以通过一个由实时时钟产生的 DMA 请求来读取它的值。

12.3.17 定时器输入异或功能

TIMx_CR2 寄存器中的 TI1S 位，允许通道 1 的输入滤波器连接到一个异或门的输出端，异或门的 3 个输入端为 TIMx_CH1、TIMx_CH2 和 TIMx_CH3。

异或输出能够被用于所有定时器的输入功能，如触发或输入捕获。下节 13.3.18 给出了此特性用于连接霍尔传感器的例子。

12.3.18 与霍尔传感器的接口

使用高级控制定时器(TIM1)产生 PWM 信号驱动马达时，可以用另一个通用 TIMx(TIM2、TIM3、TIM4)定时器作为“接口定时器”来连接霍尔传感器，见图 80，3 个定时器输入脚 (CC1、CC2、CC3)通过一个异或门连接到 TI1 输入通道(通过设置 TIMx_CR2 寄存器中的 TI1S 位 来选择)，“接口定时器”捕获这个信号。

从模式控制器被配置于复位模式，从输入是 TI1F_ED。每当 3 个输入之一变化时，计数器从新从 0 开始计数。这样产生一个由霍尔输入端的任何变化而触发的时间基准。

“接口定时器”上的捕获/比较通道 1 配置为捕获模式，捕获信号为 TRC(见图 63)。捕获值反映了两个输入变化间的时间延迟，给出了马达速度的信息。

“接口定时器”可以用来在输出模式产生一个脉冲，这个脉冲可以(通过触发一个 COM 事件)用 于改变高级定时器 TIM1 各个通道的属性，而高级控制定时器产生 PWM 信号驱动马达。因此“接口定时器”通道必须编程为在一个指定的延时(输出比较或 PWM 模式)之后产生一个正脉冲，这个脉冲通过 TRGO 输出被送到高级控制定时器 TIM1。

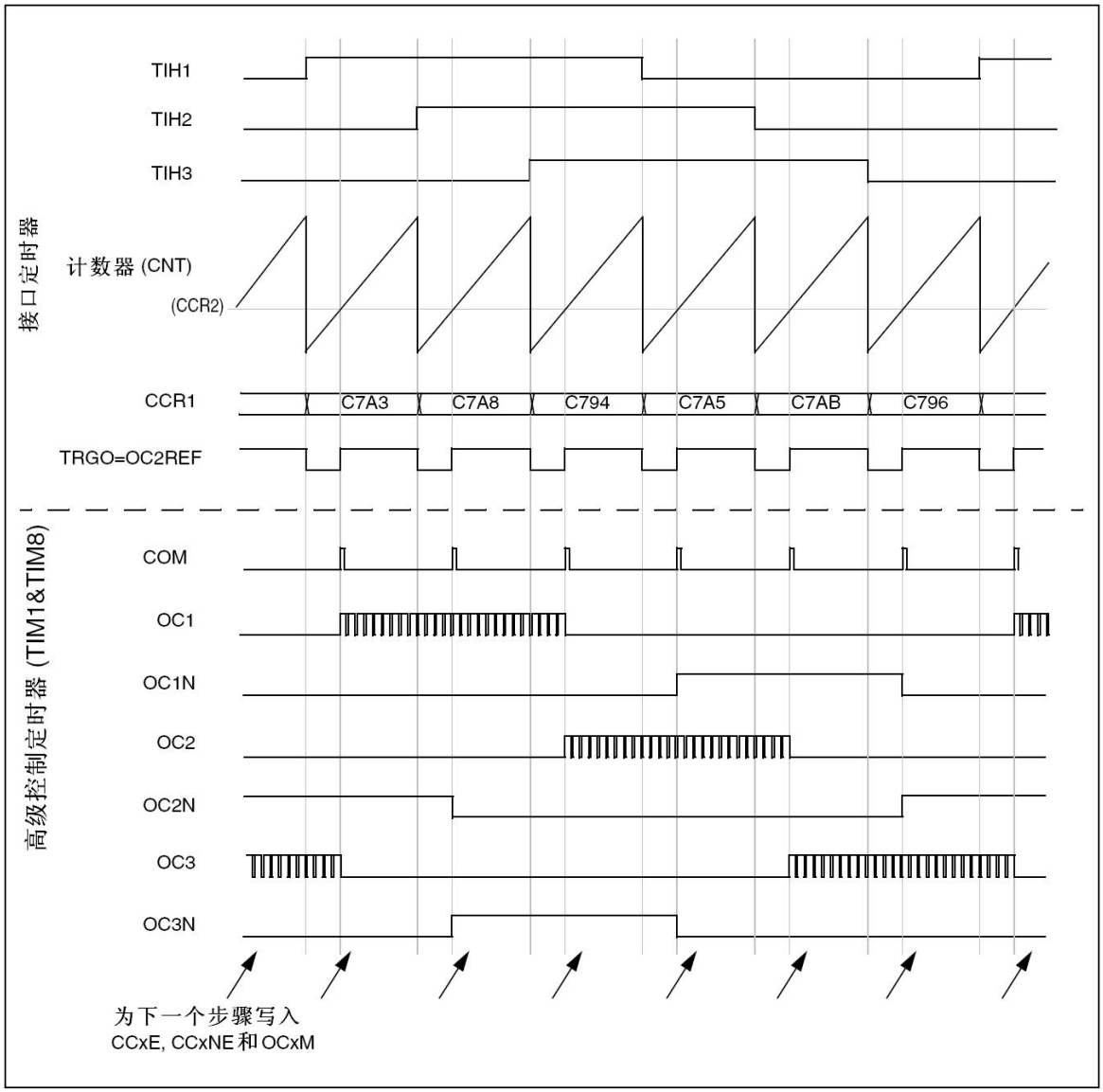
举例：霍尔输入连接到 TIMx 定时器，要求每次任一霍尔输入上发生变化之后的一个指定的时刻，改变高级控制定时器 TIMx 的 PWM 配置。

- 置 TIMx_CR2 寄存器的 TI1S 位为’1’，配置三个定时器输入逻辑或到 TI1 输入，
- 时基编程：置 TIMx_ARR 为其最大值(计数器必须通过 TI1 的变化清零)。设置预分频器得到 一个最大的计数器周期，它长于传感器上的两次变化的时间间隔。
- 设置通道 1 为捕获模式(选中 TRC)：置 TIMx_CCMR1 寄存器中 CC1S=01，如果需要，还可以设置数字滤波器。
- 设置通道 2 为 PWM2 模式，并具有要求的延时：置 TIMx_CCMR1 寄存器中的 OC2M=111 和 CC2S=00。
- 选择 OC2REF 作为 TRGO 上的触发输出：置 TIMx_CR2 寄存器中的 MMS=101。

在高级控制寄存器 TIM1 中，正确的 ITR 输入必须是触发器输入，定时器被编程为产生 PWM 信号，捕获/比较控制信号为预装载的(TIMx_CR2 寄存器中 CCPC=1)，同时触发输入控制 COM 事件(TIMx_CR2 寄存器中 CCUS=1)。在一次 COM 事件后，写入下一步的 PWM 控制位(CCxE、OCxM)，这可以在处理 OC2REF 上升沿的中断子程序里实现。

下图显示了这个实例

图 80 霍尔传感器接口的实例



12.3.19 TIMx 定时器和外部触发的同步

TIMx 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIMx_CR1 寄存器的 URS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器(TIMx_ARR，TIMx_CCRx)都被更新了。

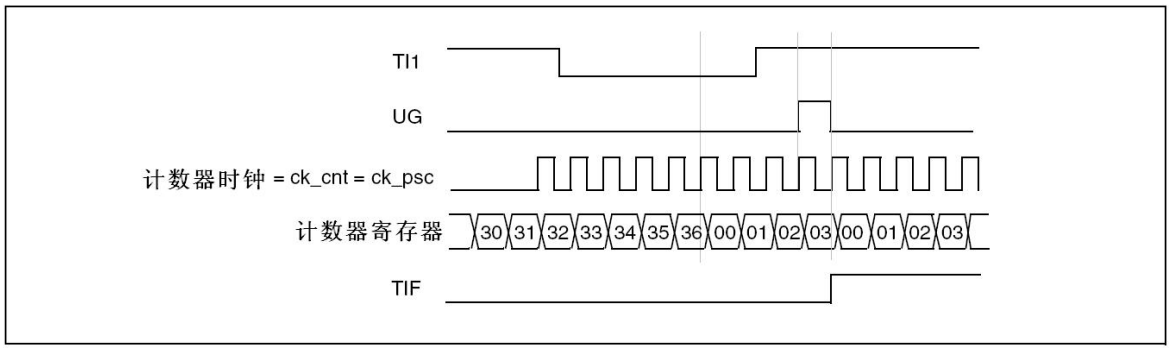
在以下的例子中，TI1 输入端的上升沿导致向上计数器被清零：

- 配置通道 1 以检测 TI1 的上升沿。配置输入滤波器的带宽(在本例中，不需要任何滤波器，因此保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)。
- 置 TIMx_SMCR 寄存器中 SMS=100，配置定时器为复位模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1 出现一个上升沿；此时，计数器被清零然后从 0 重新开始计数。同时，触发标志(TIMx_SR 寄存器中的 TIF 位)被设置，根据 TIMx_DIER 寄存器中 TIE(中断使能)位和 TDE(DMA 使能)位的设置，产生一个中断请求或一个 DMA 请求。

下图显示当自动重装载寄存器 TIMx_ARR=0x36 时的动作。在 TI1 上升沿和计数器的实际复位之间的延时取决于 TI1 输入端的重同步电路。

图 81 复位模式下的控制电路



从模式：门控模式

按照选中的输入端电平使能计数器。

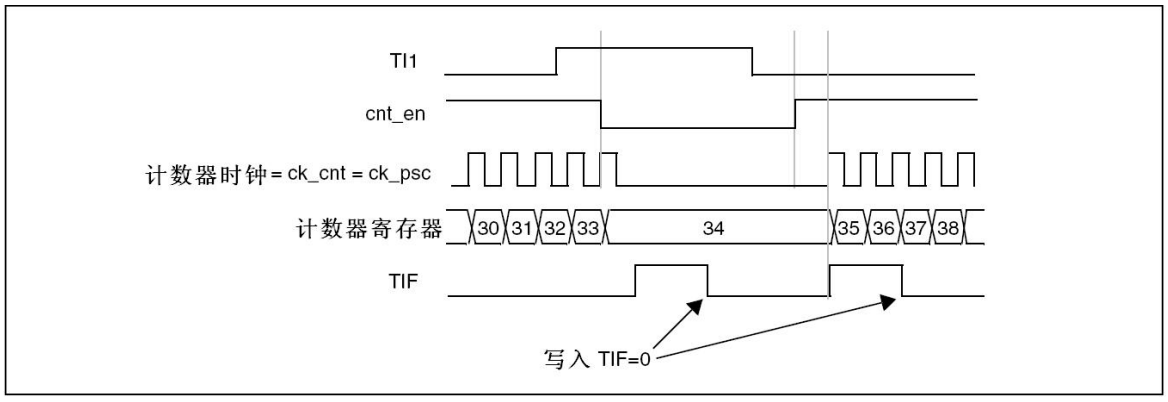
在如下的例子中，计数器只在 TI1 为低时向上计数：

- 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽(本例中，不需要滤波，所以保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=1 以确定极性(只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=0，则计数器不能 启动，不论触发输入电平如何。

只要 TI1 为低，计数器开始依据内部时钟计数，一旦 TI1 变高则停止计数。当计数器开始或停止时都设置 TIMx_SR 中的 TIF 标置。

TI1 上升沿和计数器实际停止之间的延时取决于 TI1 输入端的重同步电路。

图 82 门控模式下的控制电路



从模式：触发模式

输入端上选中的事件使能计数器。

在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

- 配置通道 2 检测 TI2 的上升沿。配置输入滤波器带宽(本例中，不需要任何滤波器，保持 IC2F=0000)。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕 获

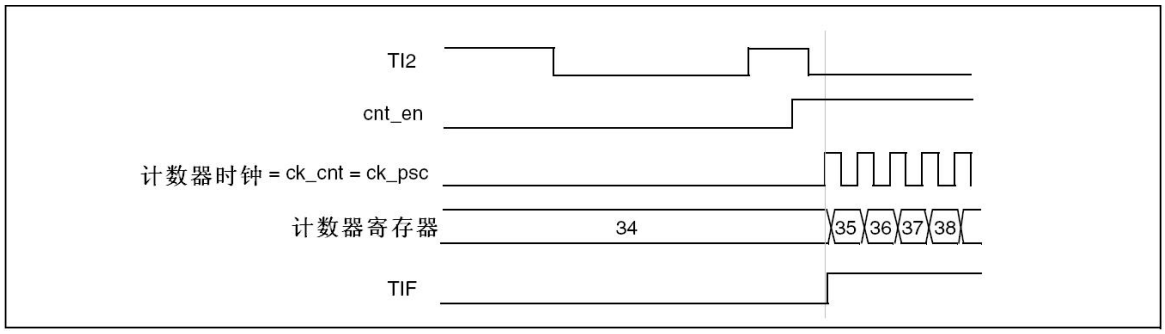
源，置 TIMx_CCMR1 寄存器中 CC2S=01。置 TIMx_CCER 寄存器中 CC2P=1 以确定极性（只检测低电平）。

- 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIMx_SMCR 寄存器中 TS=110，选择 TI2 作为输入源。

当 TI2 出现一个上升沿时，计数器开始在内部时钟驱动下计数，同时设置 TIF 标志。

TI2 上升沿和计数器启动计数之间的延时，取决于 TI2 输入端的重同步电路。

图 83 触发器模式下的控制电路



从模式：外部时钟模式 2 ＋ 触发模式

外部时钟模式 2 可以与另一种从模式(外部时钟模式 1 和编码器模式除外)一起使用。这时，ETR 信号被用作外部时钟的输入，在复位模式、门控模式或触发模式可以选择另一个输入作为触发输入。不建议使用 TIMx_SMCR 寄存器的 TS 位选择 ETR 作为 TRGI。

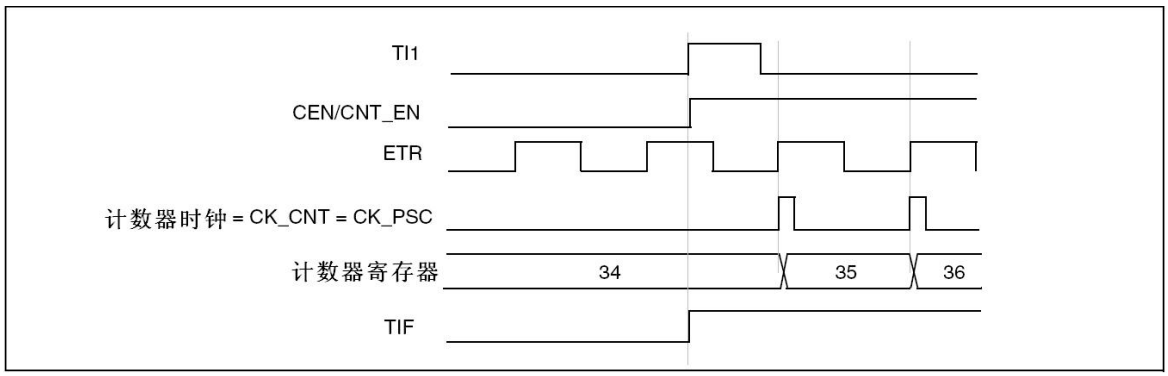
在下面的例子中，一旦在 TI1 上出现一个上升沿，计数器即在 ETR 的每一个上升沿向上计数一次：

1. 通过 TIMx_SMCR 寄存器配置外部触发输入电路：
   - ETF=0000：没有滤波
   - ETPS=00：不用预分频器
   - ETP=0：检测 ETR 的上升沿，置 ECE=1 使能外部时钟模式 2。
2. 按如下配置通道 1，检测 TI 的上升沿：
   - IC1F=0000：没有滤波
   - 触发操作中不使用捕获预分频器，不需要配置
   - 置 TIMx_CCMR1 寄存器中 CC1S=01，选择输入捕获源
   - 置 TIMx_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)
3. 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式。置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。

当 TI1 上出现一个上升沿时，TIF 标志被设置，计数器开始在 ETR 的上升沿计数。

ETR 信号的上升沿和计数器实际复位间的延时，取决于 ETRP 输入端的重同步电路。

图 84 外部时钟模式 2＋触发模式下的控制电路



12.3.20 定时器同步

所有 TIM 定时器在内部相连，用于定时器同步或链接。详见下一章 14.3.15 节

12.3.21 调试模式

当微控制器进入调试模式时(Cortex-M3 核心停止)，根据 DBG 模块中 DBG_TIMx_STOP 的设置，TIMx 计数器可以或者继续正常操作，或者停止。详见随后的 25.15.2 节。

12.4 TIM1 寄存器描述

关于在寄存器描述里面所用到的缩写，详见第 1 章。 可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

12.4.1 TIM1 控制寄存器 1(TIMx_CR1)

偏移地址：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:10	保留，始终读为 0。
位 9:8	**CKD[1:0]**：时钟分频因子(Clock division) 这 2 位定义在定时器时钟(CK_INT)频率、死区时间和由死区发生器与数字滤波器(ETR,TIx)所用的采样时钟之间的分频比例。 00：$t_{DTS} = t_{CK\_INT}$ 01：$t_{DTS} = 2 \times t_{CK\_INT}$ 10：$t_{DTS} = 4 \times t_{CK\_INT}$ 11：保留，不要使用这个配置
位 7	**ARPE**：自动重装载预装载允许位(Auto-reload preload enable) 0：TIMx_ARR 寄存器没有缓冲； 1：TIMx_ARR 寄存器被装入缓冲器。
位 6:5	**CMS[1:0]**：选择中央对齐模式(Center-aligned mode selection) 00：边沿对齐模式。计数器依据方向位(DIR)向上或向下计数。 01：中央对齐模式 1。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位，只在计数器向下计数时被设置。 10：中央对齐模式 2。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位，只在计数器向上计数时被设置。 11：中央对齐模式 3。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位，在计数器向上和向下计数时均被设置。 注：在计数器开启时(CEN=1)，不允许从边沿对齐模式转换到中央对齐模式。
位 4	**DIR**：方向(Direction) 0：计数器向上计数； 1：计数器向下计数。 注：当计数器配置为中央对齐模式或编码器模式时，该位为只读。
位 3	**OPM**：单脉冲模式(One pulse mode) 0： 在发生更新事件时，计数器不停止； 1：在发生下一次更新事件(清除 CEN 位)时，计数器停止。

位 2	**URS**：更新请求源(Update request source) 软件通过该位选择 UEV 事件的源 0：如果使能了更新中断或 DMA 请求，则下述任一事件产生更新中断或 DMA 请求： – 计数器溢出/下溢 – 设置 UG 位 – 从模式控制器产生的更新 1：如果使能了更新中断或 DMA 请求，则只有计数器溢出/下溢才产生更新中断或 DMA 请求。
位 1	**UDIS**：禁止更新(Update disable) 软件通过该位允许/禁止 UEV 事件的产生 0：允许 UEV。更新(UEV)事件由下述任一事件产生： – 计数器溢出/下溢 – 设置 UG 位 – 从模式控制器产生的更新 具有缓存的寄存器被装入它们的预装载值。(译注：更新影子寄存器) 1：禁止 UEV。不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
位 0	**CEN**：使能计数器(Counter enable) 0：禁止计数器； 1：使能计数器。 注：在软件设置了 CEN 位后，外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。

12.4.2 TIM1 控制寄存器 2(TIMx_CR2)

偏移地址：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	保留	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

位 15	保留，始终读为 0。
位 14	**OIS4**：输出空闲状态 4(OC4 输出)。参见 OIS1 位。
位 13	**OIS3N**：输出空闲状态 3(OC3N 输出)。参见 OIS1N 位。
位 12	**OIS3**：输出空闲状态 3(OC3 输出)。参见 OIS1 位。
位 11	**OIS2N**：输出空闲状态 2(OC2N 输出)。参见 OIS1N 位。
位 10	**OIS2**：输出空闲状态 2(OC2 输出)。参见 OIS1 位。
位 9	**OIS1N**：输出空闲状态 1(OC1N 输出) (Output Idle state1) 0：当 MOE=0 时，死区后 OC1N=0； 1：当 MOE=0 时，死区后 OC1N=1。 注：已经设置了 LOCK(TIMx_BKR 寄存器)级别 1、2 或 3 后，该位不能被修改。
位 8	**OIS1**：输出空闲状态 1(OC1 输出) (Output Idle state 1) 0：当 MOE=0 时，如果实现了 OC1N，则死区后 OC1=0； 1：当 MOE=0 时，如果实现了 OC1N，则死区后 OC1=1。 注：已经设置了 LOCK(TIMx_BKR 寄存器)级别 1、2 或 3 后，该位不能被修改。
位 7	**TI1S**：TI1 选择(TI1 selection) 0：TIMx_CH1 引脚连到 TI1 输入； 1：TIMx_CH1、TIMx_CH2 和 TIMx_CH3 引脚经异或后连到 TI1 输入。
位 6:4	**MMS[2:0]**：主模式选择 (Master mode selection) 这 3 位用于选择在主模式下送到从定时器的同步信息(TRGO)。可能的组合如下： 000：**复位** – TIMx_EGR 寄存器的 UG 位被用于作为触发输出(TRGO)。如果是触发输入产生的

	复位(从模式控制器处于复位模式)，则 TRGO 上的信号相对实际的复位会有一个延迟。 001：**使能** – 计数器使能信号 CNT_EN 被用于作为触发输出(TRGO)。有时需要在同一时间启动多个定时器或控制在一段时间内使能从定时器。计数器使能信号是通过 CEN 控制位和门控模式下的触发输入信号的逻辑或产生。当计数器使能信号受控于触发输入时，TRGO 上会有一个延迟，除非选择了主/从模式(见 TIMx_SMCR 寄存器中 MSM 位的描述)。 010：**更新** – 更新事件被选为触发输入(TRGO)。例如，一个主定时器的时钟可以被用作一个从定时器的预分频器。 011：**比较脉冲** – 在发生一次捕获或一次比较成功时，当要设置 CC1IF 标志时(即使它已经高)，触发输出送出一个正脉冲(TRGO)。 100：**比较**– OC1REF 信号被用于作为触发输出(TRGO)。 101：**比较**– OC2REF 信号被用于作为触发输出(TRGO)。 110：**比较**– OC3REF 信号被用于作为触发输出(TRGO)。 111：**比较**– OC4REF 信号被用于作为触发输出(TRGO)。
位 3	**CCDS**：捕获/比较的 DMA 选择(Capture/compare DMA selection) 0：当发生 CCx 事件时，送出 CCx 的 DMA 请求； 1：当发生更新事件时，送出 CCx 的 DMA 请求。
位 2	**CCUS**：捕获/比较控制更新选择(Capture/compare control update selection) 0：如果捕获/比较控制位是预装载的(CCPC=1)，只能通过设置 COM 位更新它们； 1：如果捕获/比较控制位是预装载的(CCPC=1)，可以通过设置 COM 位或 TRGI 上的一个上升沿更新它们。 注：该位只对具有互补输出的通道起作用。
位 1	保留，始终读为 0。
位 0	**CCPC**：捕获/比较预装载控制位(Capture/compare preloaded control) 0：CCxE，CCxNE 和 OCxM 位不是预装载的； 1：CCxE，CCxNE 和 OCxM 位是预装载的；设置该位后，它们只在设置了 COM 位后被更新。 注：该位只对具有互补输出的通道起作用。

12.4.3 TIM1 从模式控制寄存器(TIMx_SMCR)

偏移地址：0x08

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			保留	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

位 15	**ETP**：外部触发极性(External trigger polarity) 该位选择是用 ETR 还是 ETR 的反相来作为触发操作 0：ETR 不反相，高电平或上升沿有效； 1：ETR 被反相，低电平或下降沿有效。
位 14	**ECE**：外部时钟使能位(External clock enable) 该位启用外部时钟模式 2 0：禁止外部时钟模式 2； 1：使能外部时钟模式 2。计数器由 ETRF 信号上的任意有效边沿驱动。 注 1：设置 ECE 位与选择外部时钟模式 1 并将 TRGI 连到 ETRF(SMS=111 和 TS=111)具有相效。 注 2：下述从模式可以与外部时钟模式 2 同时使用：复位模式，门控模式和触发模式；但是，这时 TRGI 不能连到 ETRF(TS 位不能是'111')。 注 3：外部时钟模式 1 和外部时钟模式 2 同时被使能时，外部时钟的输入是 ETRF。

位 13:12	**ETPS[1:0]**：外部触发预分频(External trigger prescaler) 外部触发信号 ETRP 的频率必须最多是 TIMxCLK 频率的 1/4。当输入较快的外部时钟时，可以使 用预分频降低 ETRP 的频率。 00：关闭预分频； 01：ETRP 频率除以 2； 10：ETRP 频率除以 4； 11：ETRP 频率除以 8。
位 11:8	**ETF[3:0]**：外部触发滤波(External trigger filter) 这些位定义了对 ETRP 信号采样的频率和对 ETRP 数字滤波的带宽。实际上，数字滤波器是一个事件计数器，它记录到 N 个事件后会产生一个输出的跳变。 0000：无滤波器，以 f_{DTS} 采样 0001：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=2 0010：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=4 0011：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=8 0100：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=6 0101：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=8 0110：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=6 0111：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=8 1000：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N=6 1001：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N=8 1010：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=5 1011：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=6 1100：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=8 1101：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=5 1110：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=6 1111：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=8
位 7	**MSM**：主/从模式(Master/slave mode) 0：无作用； 1：触发输入(TRGI)上的事件被延迟了，以允许在当前定时器(通过 TRGO)与它的从定时器间的完美同步。这对要求把几个定时器同步到一个单一的外部事件时是非常有用的。
位 6:4	**TS[2:0]**：触发选择(Trigger selection) 这 3 位选择用于同步计数器的触发输入。 000：内部触发 0(ITR0) 001：内部触发 1(ITR1) 010：内部触发 2(ITR2) 011：内部触发 3(ITR3) 100：TI1 的边沿检测器(TI1F_ED) 101：滤波后的定时器输入 1(TI1FP1) 110：滤波后的定时器输入 2(TI2FP2) 111：外部触发输入(ETRF) 更多有关 ITRx 的细节，参见表 52。 注：这些位只能在未用到(如 SMS=000)时被改变，以避免在改变时产生错误的边沿检测。
位 3	保留，始终读为 0。
位 2:0	**SMS[2:0]**：从模式选择(Slave mode selection) 当选择了外部信号，触发信号(TRGI)的有效边沿与选中的外部输入极性相关(见输入控制寄存器和控制寄存器的说明) 000：关闭从模式– 如果 CEN=1，则预分频器直接由内部时钟驱动。 001：编码器模式 1 – 根据 TI1FP1 的电平，计数器在 TI2FP2 的边沿向上/下计数。 010：编码器模式 2 – 根据 TI2FP2 的电平，计数器在 TI1FP1 的边沿向上/下计数。 011：编码器模式 3 – 根据另一个信号的输入电平，计数器在 TI1FP1 和 TI2FP2 的边沿向上/下计数。 100：复位模式 – 选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。 101：门控模式 – 当触发输入(TRGI)为高时，计数器的时钟开启。一旦触发输入变为低，则计数器停止(但不复位)。计数器的启动和停止都是受控的。 110：触发模式 – 计数器在触发输入 TRGI 的上升沿启动(但不复位)，只有计数器的启动是受控的。 111：外部时钟模式 1 – 选中的触发输入(TRGI)的上升沿驱动计数器。 注：如果 TI1F_EN 被选为触发输入(TS=100)时，不要使用门控模式。这是因为，TI1F_ED 在每次 TI1F 变化时输出一个脉冲，然而门控模式是要检查触发输入的电平。

表 52 TIMx 内部触发连接

从定时器	ITR0 (TS=000)	ITR1 (TS=001)	ITR2 (TS=010)	ITR3 (TS=011)
TIM1	-	TIM2	TIM3	TIM4
-	TIM1	TIM2	TIM4	-

12.4.4 TIM1 DMA/中断使能寄存器(TIMx_DIER)

偏移地址：0x0C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15	保留，始终读为 0。
位 14	**TDE**：允许触发 DMA 请求(Trigger DMA request enable) 0：禁止触发 DMA 请求； 1：允许触发 DMA 请求。
位 13	**COMDE**：允许 COM 的 DMA 请求 (COM DMA request enable) 0：禁止 COM 的 DMA 请求； 1：允许 COM 的 DMA 请求。
位 12	**CC4DE**：允许捕获/比较 4 的 DMA 请求 (Capture/Compare 4 DMA request enable) 0：禁止捕获/比较 4 的 DMA 请求； 1：允许捕获/比较 4 的 DMA 请求。
位 11	**CC3DE**：允许捕获/比较 3 的 DMA 请求 (Capture/Compare 3 DMA request enable) 0：禁止捕获/比较 3 的 DMA 请求； 1：允许捕获/比较 3 的 DMA 请求。
位 10	**CC2DE**：允许捕获/比较 2 的 DMA 请求 (Capture/Compare 2 DMA request enable) 0：禁止捕获/比较 2 的 DMA 请求； 1：允许捕获/比较 2 的 DMA 请求。
位 9	**CC1DE**：允许捕获/比较 1 的 DMA 请求 (Capture/Compare 1 DMA request enable) 0：禁止捕获/比较 1 的 DMA 请求； 1：允许捕获/比较 1 的 DMA 请求。
位 8	**UDE**：允许更新的 DMA 请求(Update DMA request enable) 0：禁止更新的 DMA 请求； 1：允许更新的 DMA 请求。
位 7	**BIE**：允许刹车中断(Break interrupt enable) 0：禁止刹车中断； 1：允许刹车中断。
位 6	**TIE**：触发中断使能(Trigger interrupt enable) 0：禁止触发中断； 1：使能触发中断。
位 5	**COMIE**：允许 COM 中断(COM interrupt enable) 0：禁止 COM 中断； 1：允许 COM 中断。
位 4	**CC4IE**：允许捕获/比较 4 中断(Capture/Compare 4 interrupt enable) 0：禁止捕获/比较 4 中断； 1：允许捕获/比较 4 中断。

位 3	**CC3IE**：允许捕获/比较 3 中断(Capture/Compare 3 interrupt enable) 0：禁止捕获/比较 3 中断； 1：允许捕获/比较 3 中断。
位 2	**CC2IE**：允许捕获/比较 2 中断(Capture/Compare 2 interrupt enable) 0：禁止捕获/比较 2 中断； 1：允许捕获/比较 2 中断。
位 1	**CC1IE**：允许捕获/比较 1 中断(Capture/Compare 1 interrupt enable) 0：禁止捕获/比较 1 中断； 1：允许捕获/比较 1 中断。
位 0	**UIE**：允许更新中断(Update interrupt enable) 0：禁止更新中断； 1：允许更新中断。

12.4.5 TIM1 状态寄存器(TIMx_SR)

偏移地址：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			CC4OF	CC3OF	CC2OF	CC1OF	保留	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc w0	rc w0	rc w0	rc w0		rc w0	rc w0	rc w0	rc w0	rc w0	rc w0	rc w0	rc w0

位 15:13	保留，始终读为 0。
位 12	**CC4OF**：捕获/比较 4 重复捕获标记(Capture/Compare 4 overcapture flag) 参见 CC1OF 描述。
位 11	**CC3OF**：捕获/比较 3 重复捕获标记(Capture/Compare 3 overcapture flag) 参见 CC1OF 描述。
位 10	**CC2OF**：捕获/比较 2 重复捕获标记(Capture/Compare 2 overcapture flag) 参见 CC1OF 描述。
位 9	**CC1OF**：捕获/比较 1 重复捕获标记 (Capture/Compare 1 overcapture flag) 仅当相应的通道被配置为输入捕获时，该标记可由硬件置 1。写 0 可清除该位。 0：无重复捕获产生； 1：计数器的值被捕获到 TIMx_CCR1 寄存器时，CC1IF 的状态已经为'1'。
位 8	保留，始终读为 0。
位 7	**BIF**：刹车中断标记(Break interrupt flag) 一旦刹车输入有效，由硬件对该位置'1'。如果刹车输入无效，则该位可由软件清'0'。 0：无刹车事件产生； 1：刹车输入上检测到有效电平。
位 6	**TIF**：触发器中断标记(Trigger interrupt flag) 当发生触发事件(当从模式控制器处于除门控模式外的其它模式时，在 TRGI 输入端检测到有效边沿，或门控模式下的任一边沿)时由硬件对该位置'1'。它由软件清'0'。 0：无触发器事件产生； 1：触发中断等待响应。
位 5	**COMIF**：COM 中断标记(COM interrupt flag) 一旦产生 COM 事件(当捕获/比较控制位：CCxE、CCxNE、OCxM 已被更新)该位由硬件置'1'。它由软件清'0'。 0：无 COM 事件产生； 1：COM 中断等待响应。
位 4	**CC4IF**：捕获/比较 4 中断标记(Capture/Compare 4 interrupt flag) 参考 CC1IF 描述。

位 3	**CC3IF**：捕获/比较 3 中断标记(Capture/Compare 3 interrupt flag) 参考 CC1IF 描述。
位 2	**CC2IF**：捕获/比较 2 中断标记(Capture/Compare 2 interrupt flag) 参考 CC1IF 描述。
位 1	**CC1IF**：捕获/比较 1 中断标记(Capture/Compare 1 interrupt flag) **如果通道 CC1 配置为输出模式：** 当计数器值与比较值匹配时该位由硬件置 1，但在中心对称模式下除外(参考 TIMx_CR1 寄存器的 CMS 位)。它由软件清'0'。 0：无匹配发生； 1：TIMx_CNT 的值与 TIMx_CCR1 的值匹配。 当 TIMx_CCR1 的内容大于 TIMx_APR 的内容时，在向上或向上/下计数模式时计数器溢出，或向下计数模式时的计数器下溢条件下，CC1IF 位变高 **如果通道 CC1 配置为输入模式：** 当捕获事件发生时该位由硬件置'1'，它由软件清'0'或通过读 TIMx_CCR1 清'0'。 0：无输入捕获产生； 1：计数器值已被捕获(拷贝)至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)。
位 0	**UIF**：更新中断标记(Update interrupt flag) 当产生更新事件时该位由硬件置'1'。它由软件清'0'。 0：无更新事件产生； 1：更新中断等待响应。当寄存器被更新时该位由硬件置'1'： – 若 TIMx_CR1 寄存器的 UDIS=0，当重复计数器数值上溢或下溢时(重复计数器=0 时产生新事件)。 – 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当设置 TIMx_EGR 寄存器的 UG=1 时产生更新件，通过软件对计数器 CNT 重新初始化时。 – 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当计数器 CNT 被触发事件重新初始化时。(参 12.4.3: TIM1 从模式控制寄存器(TIMx_SMCR))。

12.4.6 TIM1 事件产生寄存器(TIMx_EGR)

偏移地址:0x14

复位值:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
								W	W	W	W	W	W	W	W

位 15:8	保留，始终读为 0。
位 7	**BG**：产生刹车事件(Break generation) 该位由软件置'1'，用于产生一个刹车事件，由硬件自动清'0'。 0：无动作； 1：产生一个刹车事件。此时 MOE=0、BIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。
位 6	**TG**：产生触发事件(Trigger generation) 该位由软件置'1'，用于产生一个触发事件，由硬件自动清'0'。 0：无动作； 1：TIMx_SR 寄存器的 TIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。
位 5	**COMG**：捕获/比较事件，产生控制更新(Capture/Compare control update generation) 该位由软件置'1'，由硬件自动清'0'。 0：无动作； 1：当 CCPC=1，允许更新 CCxE、CCxNE、OCxM 位。 **注**：该位只对拥有互补输出的通道有效。
位 4	**CC4G**：产生捕获/比较 4 事件(Capture/Compare 4 generation) 参考 CC1G 描述。
位 3	**CC3G**：产生捕获/比较 3 事件(Capture/Compare 3 generation) 参考 CC1G 描述。
位 2	**CC2G**：产生捕获/比较 2 事件(Capture/Compare 2 generation) 参考 CC1G 描述。
位 1	**CC1G**：产生捕获/比较 1 事件(Capture/Compare 1 generation) 该位由软件置'1'，用于产生一个捕获/比较事件，由硬件自动清'0'。 0：无动作； 1：在通道 CC1 上产生一个捕获/比较事件： **若通道 CC1 配置为输出：** 设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。 **若通道 CC1 配置为输入：** 当前的计数器值被捕获至 TIMx_CCR1 寄存器；设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。若 CC1IF 已经为 1，则设置 CC1OF=1。
位 0	**UG**：产生更新事件(Update generation) 该位由软件置'1'，由硬件自动清'0'。 0：无动作； 1：重新初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清'0'（但是预分频系数不变）。若在中心对称模式下或 DIR=0(向上计数)则计数器被清'0'；若 DIR=1(向下计数)则计数器取 TIMx_ARR 的值。

12.4.7 TIM1 捕获/比较模式寄存器 1(TIMx_CCMR1)

偏移地址：0x18

复位值：0x0000

通道可用于输入(捕获模式)或输出(比较模式)，通道的方向由相应的 CCxS 位定义。该寄存器其它位的作用在输入和输出模式下不同。OCxx 描述了通道在输出模式下的功能，ICxx 描述了通道在输入模式下的功能。因此必须注意，同一个位在输出模式和输入模式下的功能是不同的。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

输出比较模式：

位 15	**OC2CE**：输出比较 2 清 0 使能(Output Compare 2 clear enable)
位 14:12	**OC2M[2:0]**：输出比较 2 模式(Output Compare 2 mode)
位 11	**OC2PE**：输出比较 2 预装载使能(Output Compare 2 preload enable)
位 10	**OC2FE**：输出比较 2 快速使能(Output Compare 2 fast enable)
位 9:8	**CC2S[1:0]**：捕获/比较 2 选择。(Capture/Compare 2 selection) 该位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC2 通道被配置为输出； 01：CC2 通道被配置为输入，IC2 映射在 TI2 上； 10：CC2 通道被配置为输入，IC2 映射在 TI1 上； 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 **注：**CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E=0)才是可写的。
位 7	**OC1CE**：输出比较 1 清'0'使能(Output Compare 1 clear enable) 0：OC1REF 不受 ETRF 输入的影响； 1：一旦检测到 ETRF 输入高电平，清除 OC1REF=0。
位 6:4	**OC1M[2:0]**：输出比较 1 模式(Output Compare 1 mode) 该 3 位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1、OC1N 的值。OC1REF 是高电平有效，而 OC1、OC1N 的有效电平取决于 CC1P、CC1NP 位。 000：冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用； 001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存(TIMx_CCR1)相同时，强制 OC1REF 为高。 010：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存(TIMx_CCR1)相同时，强制 OC1REF 为低。 011：翻转。当 TIMx_CCR1=TIMx_CNT 时，翻转 OC1REF 的电平。 100：强制为无效电平。强制 OC1REF 为低。 101：强制为有效电平。强制 OC1REF 为高。 110：PWM 模式 1－在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为有效电平，否则无效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为无效电平(OC1REF=0)，否则为有效电平(OC1REF=1)。 111：PWM 模式 2－在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为无效电平，否则有效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为有效电平，否则为无效平。 注 1：一旦 LOCK 级别设为 3(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC1S=00(该通道配置出)则该位不能被修改。 注 2：在 PWM 模式 1 或 PWM 模式 2 中，只有当比较结果改变了或在输出比较模式中从冻结切换到 PWM 模式时，OC1REF 电平才改变。
位 3	**OC1PE**：输出比较 1 预装载使能(Output Compare 1 preload enable)

	0：禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器，并且新写入的值立即起作用。 1：开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 预装载值在更新事件到来时被加载至当前寄存器中。 注 1：一旦 LOCK 级别设为 3(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC1S=00(该通道配置出)则该位不能被修改。 注 2：仅在单脉冲模式下(TIMx_CR1 寄存器的 OPM=1)，可以在未确认预装载寄存器情况下使用 PWM 模式，否则其动作不确定。
位 2	**OC1FE**：输出比较 1 快速使能(Output Compare 1 fast enable) 该位用于加快 CC 输出对触发输入事件的响应。 0：根据计数器与 CCR1 的值，CC1 正常操作，即使触发器是打开的。当触发器的输入有一个有效沿时，激活 CC1 输出的最小延时为 5 个时钟周期。 1：输入到触发器的有效沿的作用就象发生了一次比较匹配。因此，OC 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CC1 输出间的延时被缩短为 3 个时钟周期。 OCFE 只在通道被配置成 PWM1 或 PWM2 模式时起作用。
位 1:0	**CC1S[1:0]**：捕获/比较 1 选择。(Capture/Compare 1 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：CC1 通道被配置为输入，IC1 映射在 TI2 上； 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0)才是可写的。

输入捕获模式：

位 15:12	**IC2F[3:0]**：输入捕获 2 滤波器(Input capture 2 filter)
位 11:10	**IC2PSC[1:0]**：输入/捕获 2 预分频器(Input capture 2 prescaler)
位 9:8	**CC2S[1:0]**：捕获/比较 2 选择(Capture/Compare 2 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC2 通道被配置为输出； 01：CC2 通道被配置为输入，IC2 映射在 TI2 上； 10：CC2 通道被配置为输入，IC2 映射在 TI1 上； 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。注：CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E=0)才是可写的。
位 7:4	**IC1F[3:0]**：输入捕获 1 滤波器(Input capture 1 filter) 这几位定义了 TI1 输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记录到 N 个事件后会产生一个输出的跳变： 0000：无滤波器，以 f_{DTS} 采样 0001：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=2 0010：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=4 0011：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=8 0100：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=6 0101：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=8 0110：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=6 0111：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=8 1000：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N=6 1001：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N=8 1010：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=5 1011：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=6 1100：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=8 1101：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=5 1110：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=6 1111：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=8

位 3:2	**IC1PSC[1:0]**：输入/捕获 1 预分频器(Input capture 1 prescaler) 这 2 位定义了 CC1 输入(IC1)的预分频系数。 一旦 CC1E=0(TIMx_CCER 寄存器中)，则预分频器复位。 00：无预分频器，捕获输入口上检测到的每一个边沿都触发一次捕获； 01：每 2 个事件触发一次捕获； 10：每 4 个事件触发一次捕获； 11：每 8 个事件触发一次捕获。
位 1:0	**CC1S[1:0]**：捕获/比较 1 选择(Capture/Compare 1 Selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：CC1 通道被配置为输入，IC1 映射在 TI2 上； 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 **注**：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0)才是可写的。

12.4.8 TIM1 捕获/比较模式寄存器 2(TIMx_CCMR2)

偏移地址：0x1C

复位值：0x0000

参看以上 CCMR1 寄存器的描述

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

输出比较模式：

位 15	**OC4CE**：输出比较 4 清 0 使能(Output compare 4 clear enable)
位 14:12	**OC4M[2:0]**：输出比较 4 模式(Output compare 4 mode)
位 11	**OC4PE**：输出比较 4 预装载使能(Output compare 4 preload enable)
位 10	**OC4FE**：输出比较 4 快速使能(Output compare 4 fast enable)
位 9:8	**CC4S[1:0]**：捕获/比较 4 选择(Capture/Compare 4 selection) 该 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC4 通道被配置为输出； 01：CC4 通道被配置为输入，IC4 映射在 TI4 上； 10：CC4 通道被配置为输入，IC4 映射在 TI3 上； 11：CC4 通道被配置为输入，IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 **注**：CC4S 仅在通道关闭时(TIMx_CCER 寄存器的 CC4E=0)才是可写的。
位 7	**OC3CE**：输出比较 3 清 0 使能(Output compare 3 clear enable)
位 6:4	**OC3M[2:0]**：输出比较 3 模式(Output compare 3 mode)
位 3	**OC3PE**：输出比较 3 预装载使能(Output compare 3 preload enable)
位 2	**OC3FE**：输出比较 3 快速使能(Output compare 3 fast enable)
位 1:0	**CC3S[1:0]**：捕获/比较 3 选择(Capture/Compare 3 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC3 通道被配置为输出； 01：CC3 通道被配置为输入，IC3 映射在 TI3 上；

	10：CC3 通道被配置为输入，IC3 映射在 TI4 上； 11：CC3 通道被配置为输入，IC3 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC3S 仅在通道关闭时(TIMx_CCER 寄存器的 CC3E=0)才是可写的。

输入捕获模式：

位 15:12	**IC4F[3:0]**：输入捕获 4 滤波器(Input capture 4 filter)
位 11:10	**IC4PSC[1:0]**：输入/捕获 4 预分频器(Input capture 4 prescaler)
位 9:8	**CC4S[1:0]**：捕获/比较 4 选择(Capture/Compare 4 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC4 通道被配置为输出； 01：CC4 通道被配置为输入，IC4 映射在 TI4 上； 10：CC4 通道被配置为输入，IC4 映射在 TI3 上； 11：CC4 通道被配置为输入，IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC4S 仅在通道关闭时(TIMx_CCER 寄存器的 CC4E=0)才是可写的。
位 7:4	**IC3F[3:0]**：输入捕获 3 滤波器(Input capture 3 filter)
位 3:2	**IC3PSC[1:0]**：输入/捕获 3 预分频器(Input capture 3 prescaler)
位 1:0	**CC3S[1:0]**：捕获/比较 3 选择(Capture/compare 3 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC3 通道被配置为输出； 01：CC3 通道被配置为输入，IC3 映射在 TI3 上； 10：CC3 通道被配置为输入，IC3 映射在 TI4 上； 11：CC3 通道被配置为输入，IC3 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC3S 仅在通道关闭时(TIMx_CCER 寄存器的 CC3E=0)才是可写的。

12.4.9 TIM1 捕获/比较使能寄存器(TIMx_CCER)

偏移地址：0x20

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:14	保留，始终读为 0。
位 13	**CC4P**：输入/捕获 4 输出极性(Capture/Compare 4 output polarity) 参考 CC1P 的描述。
位 12	**CC4E**：输入/捕获 4 输出使能(Capture/Compare 4 output enable) 参考 CC1E 的描述。
位 11	**CC3NP**：输入/捕获 3 互补输出极性(Capture/Compare 3 complementary output polarity) 参考 CC1NP 的描述。
位 10	**CC3NE**：输入/捕获 3 互补输出使能(Capture/Compare 3 complementary output enable) 参考 CC1NE 的描述。
位 9	**CC3P**：输入/捕获 3 输出极性(Capture/Compare 3 output polarity) 参考 CC1P 的描述。
位 8	**CC3E**：输入/捕获 3 输出使能(Capture/Compare 3 output enable) 参考 CC1E 的描述。

位 7	**CC2NP**：输入/捕获 2 互补输出极性(Capture/Compare 2 complementary output polarity) 参考 CC1NP 的描述。
位 6	**CC2NE**：输入/捕获 2 互补输出使能(Capture/Compare 2 complementary output enable) 参考 CC1NE 的描述。
位 5	**CC2P**：输入/捕获 2 输出极性(Capture/Compare 2 output polarity) 参考 CC1P 的描述。
位 4	**CC2E**：输入/捕获 2 输出使能(Capture/Compare 2 output enable) 参考 CC1E 的描述。
位 3	**CC1NP**：输入/捕获 1 互补输出极性(Capture/Compare 1 complementary output polarity) 0：OC1N 高电平有效； 1：OC1N 低电平有效。 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 3 或 2 且 CC1S=00(通道配置为输出) 则该位不能被修改。
位 2	**CC1NE**：输入/捕获 1 互补输出使能(Capture/Compare 1 complementary output enable) 0：关闭－ OC1N 禁止输出，因此 OC1N 的电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和CC1E 位的值。 1：开启－ OC1N 信号输出到对应的输出引脚，其输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位的值。
位 1	**CC1P**：输入/捕获 1 输出极性(Capture/Compare 1 output polarity) **CC1 通道配置为输出：** 0：OC1 高电平有效； 1：OC1 低电平有效。 **CC1 通道配置为输入：** 该位选择是 IC1 还是 IC1 的反相信号作为触发或捕获信号。 0：不反相：捕获发生在 IC1 的上升沿；当用作外部触发器时，IC1 不反相。 1：反相：捕获发生在 IC1 的下降沿；当用作外部触发器时，IC1 反相。 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 3 或 2，则该位不能被修改。
位 0	**CC1E**：输入/捕获 1 输出使能(Capture/Compare 1 output enable) **CC1 通道配置为输出：** 0：关闭－ OC1 禁止输出，因此 OC1 的输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 位的值。 1：开启－ OC1 信号输出到对应的输出引脚，其输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 位的值。 **CC1 通道配置为输入：** 该位决定了计数器的值是否能捕获入 TIMx_CCR1 寄存器。 0：捕获禁止； 1：捕获使能。

表 53 带刹车功能的互补输出通道 OCx 和 OCxN 的控制位

控制位					输出状态(1)	
MOE 位	OSSI 位	OSSR 位	CCxE 位	CCxNE 位	OCx 输出状态	OCxN 输出状态
1	X	0	0	0	输出禁止(与定时器断开) OCx=0，OCx_EN=0	输出禁止(与定时器断开) OCxN=0，OCxN_EN=0
		0	0	1	输出禁止(与定时器断开) OCx=0，OCx_EN=0	OCxREF + 极性， OCxN= OCxREF xor CCxNP， OCxN_EN=1
		0	1	0	OCxREF + 极性， OCx= OCxREF xor CCxP， OCx_EN=1	输出禁止(与定时器断开) OCxN=0，OCxN_EN=0
		0	1	1	OCxREF + 极性+ 死区， OCx_EN=1	OCxREF 反相+ 极性+ 死区， OCxN_EN=1
		1	0	0	输出禁止(与定时器断开) OCx=CCxP，OCx_EN=0	输出禁止(与定时器断开) OCxN=CCxNP，OCxN_EN=0
		1	0	1	关闭状态(输出使能且为无效电平) OCx=CCxP， OCx_EN=1	OCxREF + 极性， OCxN= OCxREF xor CCxNP， OCxN_EN=1
		1	1	0	OCxREF + 极性， OCx= OCxREF xor CCxP， OCx_EN=1	关闭状态(输出使能且为无效电平) OCxN=CCxNP， OCxN_EN=1
		1	1	1	OCxREF + 极性+ 死区， OCx_EN=1	OCxREF 反相+ 极性+ 死区， OCxN_EN=1
0	0	X	0	0	输出禁止(与定时器断开) 异步地：OCx=CCxP，OCx_EN=0，OCxN=CCxNP，OCxN_EN=0； 若时钟存在：经过一个死区时间后 OCx=OISx，OCxN=OISxN，假设 OISx 与 OISxN 并不都对应 OCx 和 OCxN 的有效电平。	
	0		0	1		
	0		1	0		
	0		1	1		
	1		0	0	关闭状态(输出使能且为无效电平) 异步地：OCx=CCxP，OCx_EN=1，OCxN=CCxNP，OCxN_EN=1； 若时钟存在：经过一个死区时间后 OCx=OISx，OCxN=OISxN，假设 OISx 与 OISxN 并不都对应 OCx 和 OCxN 的有效电平。	
	1		0	1		
	1		1	0		
	1		1	1		

1. 如果一个通道的 2 个输出都没有使用(CCxE = CCxNE = 0)，那么 OISx, OISxN, CCxP 和 CCxNP 都必须清零。

注：引脚连接到互补的 OCx 和 OCxN 通道的外部 I/O 引脚的状态，取决于 OCx 和 OCxN 通道状态和 GPIO 以及 AFIO 寄存器。

12.4.10 TIM1 计数器(TIMx_CNT)

偏移地址：0x24

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0	**CNT[15:0]**：计数器的值(Counter value)

12.4.11 TIM1 预分频器(TIMx_PSC)

偏移地址：0x28

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0	**PSC[15:0]**：预分频器的值(Prescaler value) 计数器的时钟频率(CK_CNT)等于 f_{CK_PSC}/(PSC[15:0]+1)。 PSC 包含了每次当更新事件产生时，装入当前预分频器寄存器的值；更新事件包括计数器被 TIM_EGR 的 UG 位清'0'或被工作在复位模式的从控制器清'0'。

12.4.12 TIM1 自动重装载寄存器(TIMx_ARR)

偏移地址:0x2C

复位值:0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0	**ARR[15:0]**: 自动重装载的值(Prescaler value) ARR 包含了将要装载入实际的自动重装载寄存器的值。 详细参考 13.3.1 节：有关 ARR 的更新和动作。 当自动重装载的值为空时，计数器不工作。

12.4.13 TIM1 重复计数寄存器(TIMx_RCR)

偏移地址：0x30

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

位 15:8	保留，始终读为 0。
位 7:0	**REP[7:0]**: 重复计数器的值(Repetition counter value) 开启了预装载功能后，这些位允许用户设置比较寄存器的更新速率(即周期性地从预装载寄存器传输到当前寄存器)；如果允许产生更新中断，则会同时影响产生更新中断的速率。 每次向下计数器 REP_CNT 达到 0，会产生一个更新事件并且计数器 REP_CNT 重新从 REP 值始计数。由于 REP_CNT 只有在周期更新事件 U_RC 发生时才重载 REP 值，因此对 TIMx_RCR 存器写入的新值只在下次周期更新事件发生时才起作用。 这意味着在 PWM 模式中，(REP+1)对应着： — 在边沿对齐模式下，PWM 周期的数目； — 在中心对称模式下，PWM 半周期的数目；

12.4.14 TIM1 捕获/比较寄存器 1(TIMx_CCR1)

偏移地址：0x34

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
Rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

位 15:0	**CCR1[15:0]**: 捕获/比较通道 1 的值(Capture/Compare 1 value) **若 CC1 通道配置为输出：** CCR1 包含了装入当前捕获/比较 1 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC1PE 位)中未选择预装载功能，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 1 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC1 端口上产生输出信号。 **若 CC1 通道配置为输入：** CCR1 包含了由上一次输入捕获 1 事件(IC1)传输的计数器值。TIMx_CCR1 只能读不能被编程。

12.4.15 TIM1 捕获/比较寄存器 2(TIMx_CCR2)

偏移地址：0x38

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

位 15:0	**CCR2[15:0]**: 捕获/比较通道 2 的值(Capture/Compare 2 value) **若 CC2 通道配置为输出：** CCR2 包含了装入当前捕获/比较 2 寄存器的值(预装载值)。 如果在 TIMx_CCMR2 寄存器(OC2PE 位)中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 2 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC2 端口上产生输出信号。 **若 CC2 通道配置为输入：** CCR2 包含了由上一次输入捕获 2 事件(IC2)传输的计数器值。TIMx_CCR2 只能读不能被编程

12.4.16 TIM1 捕获/比较寄存器 3(TIMx_CCR3)

偏移地址：0x3C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

位 15:0	**CCR3[15:0]**: 捕获/比较通道 3 的值(Capture/Compare 3 value) **若 CC3 通道配置为输出：** CCR3 包含了装入当前捕获/比较 3 寄存器的值(预装载值)。 如果在 TIMx_CCMR3 寄存器(OC3PE 位)中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 3 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC3 端口上产生输出信号。 **若 CC3 通道配置为输入：** CCR3 包含了由上一次输入捕获 3 事件(IC3)传输的计数器值。TIMx_CCR3 只能读不能被编程

12.4.17 TIM1 捕获/比较寄存器(TIMx_CCR4)

偏移地址：0x40

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

位 15:0	**CCR4[15:0]**: 捕获/比较通道 4 的值(Capture/Compare 4 value) **若 CC4 通道配置为输出：** CCR4 包含了装入当前捕获/比较 4 寄存器的值(预装载值)。 如果在 TIMx_CCMR4 寄存器(OC4PE 位)中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 4 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC4 端口上产生输出信号。 **若 CC4 通道配置为输入：** CCR4 包含了由上一次输入捕获 4 事件(IC4)传输的计数器值。TIMx_CCR4 只能读不能被编程

12.4.18 TIM1 刹车和死区寄存器(TIMx_BDTR)

偏移地址：0x44

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

注释： *根据锁定设置，AOE、BKP、BKE、OSSI、OSSR 和 DTG[7:0]位均可被写保护，有必要在第一次写入 TIMx_BDTR 寄存器时对它们进行配置。*

位 15	**MOE**: 主输出使能(Main output enable) 一旦刹车输入有效，该位被硬件异步清'0'。根据 AOE 位的设置值，该位可以由软件清'0'或被自动置 1。它仅对配置为输出的通道有效。 0：禁止 OC 和 OCN 输出或强制为空闲状态； 1：如果设置了相应的使能位(TIMx_CCER 寄存器的 CCxE、CCxNE 位)，则开启 OC 和 OCN 出。 有关 OC/OCN 使能的细节，参见 13.4.9 节，TIM1 捕获/比较使能寄存器(TIMx_CCER)。
位 14	**AOE**: 自动输出使能(Automatic output enable) 0：MOE 只能被软件置'1'； 1：MOE 能被软件置'1'或在下一个更新事件被自动置'1'(如果刹车输入无效)。 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为'1'，则该位不能被修改。
位 13	**BKP**: 刹车输入极性(Break polarity) 0：刹车输入低电平有效； 1：刹车输入高电平有效。 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为'1'，则该位不能被修改。 注：任何对该位的写操作都需要一个 APB 时钟的延迟以后才能起作用。
位 12	**BKE**: 刹车功能使能(Break enable) 0：禁止刹车输入(BRK 及 CCS 时钟失效事件)； 1：开启刹车输入(BRK 及 CCS 时钟失效事件)。 注：当设置了 LOCK 级别 1 时(TIMx_BDTR 寄存器中的 LOCK 位)，该位不能被修改。 注：任何对该位的写操作都需要一个 APB 时钟的延迟以后才能起作用。
位 11	**OSSR**: 运行模式下“关闭状态”选择(Off-state selection for Run mode) 该位用于当 MOE=1 且通道为互补输出时。没有互补输出的定时器中不存在 OSSR 位。 参考 OC/OCN 使能的详细说明(13.4.9 节，TIM1 捕获/比较使能寄存器(TIMx_CCER))。 0：当定时器不工作时，禁止 OC/OCN 输出(OC/OCN 使能输出信号=0)； 1：当定时器不工作时，一旦 CCxE=1 或 CCxNE=1，首先开启 OC/OCN 并输出无效电平，然后置 OC/OCN 使能输出信号=1。 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 2，则该位不能被修改。
位 10	**OSSI**: 空闲模式下“关闭状态”选择(Off-state selection for Idle mode) 该位用于当 MOE=0 且通道设为输出时。 参考 OC/OCN 使能的详细说明(13.4.9 节，TIM1 捕获/比较使能寄存器(TIMx_CCER))。 0：当定时器不工作时，禁止 OC/OCN 输出(OC/OCN 使能输出信号=0)； 1：当定时器不工作时，一旦 CCxE=1 或 CCxNE=1，OC/OCN 首先输出其空闲电平，然后 OC/OCN 使能输出信号=1。 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 2，则该位不能被修改。
位 9:8	**LOOK[1:0]**: 锁定设置(Lock configuration) 该位为防止软件错误而提供写保护。 00：锁定关闭，寄存器无写保护； 01：锁定级别 1，不能写入 TIMx_BDTR 寄存器的 DTG、BKE、BKP、AOE 位和 TIMx_CR2 寄存器的 OISx/OISxN 位； 10：锁定级别 2，不能写入锁定级别 1 中的各位，也不能写入 CC 极性位(一旦相关通道 CCxS 位设为输出，CC 极性位是 TIMx_CCER 寄存器的 CCxP/CCNxP 位)以及 OSSR/OSSI 位； 11：锁定级别 3，不能写入锁定级别 2 中的各位，也不能写入 CC 控制位(一旦相关通道 CCxS 位设为输出，CC 控制位是 TIMx_CCMRx 寄存器的 OCxM/OCxPE 位)； 注：在系统复位后，只能写一次 LOCK 位，一旦写入 TIMx_BDTR 寄存器，则其内容冻结直至复位。

位 7:0	**UTG[7:0]**: 死区发生器设置(Dead-time generator setup) 这些位定义了插入互补输出之间的死区持续时间。假设 DT 表示其持续时间： DTG[7:5]=0xx => DT=DTG[7:0] × T_{dtg}，T_{dtg} = T_{DTS}； DTG[7:5]=10x => DT=(64+DTG[5:0]) × T_{dtg}，T_{dtg} = 2 × T_{DTS}； DTG[7:5]=110 => DT=(32+DTG[4:0]) × T_{dtg}，T_{dtg} = 8 × T_{DTS}； DTG[7:5]=111 => DT=(32+DTG[4:0])× T_{dtg}，T_{dtg} = 16 × T_{DTS}； 例：若 T_{DTS} = 125ns(8MHZ)，可能的死区时间为： 0 到 15875ns，若步长时间为 125ns； 16us 到 31750ns，若步长时间为 250ns； 32us 到 63us，若步长时间为 1us； 64us 到 126us，若步长时间为 2us； 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 1、2 或 3，则不能修改这些位。

12.4.19 TIM1 DMA 控制寄存器(TIMx_DCR)

偏移地址：0x48

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			DBL[4:0]					保留			DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

位 15:13	保留，始终读为 0
位 12:8	**DBL[4:0]**: DMA 连续传送长度(DMA burst length) 这些位定义了 DMA 在连续模式下的传送长度(当对 TIMx_DMAR 寄存器进行读或写时，定时器则进行一次连续传送)，即：定义传输的次数，传输可以是半字(双字节)或字节： 00000：1 次传输　　00001：2 次传输 00010：3 次传输　　..... .....　　10001：18 次传输 例：我们考虑这样的传输：DBL=7，DBA=TIM2_CR1 - 如果 DBL=7，DBA=TIM2_CR1 表示待传输数据的地址，那么传输的地址由下式给出： (TIMx_CR1 的地址) + DBA + (DMA 索引), 其中DMA 索引= DBL 其中(TIMx_CR1 的地址) + DBA 再加上 7，给出了将要写入或者读出数据的地址，这样数据的传输将发生在从地址(TIMx_CR1 的地址) + DBA 开始的 7 个寄存器。 根据 DMA 数据长度的设置，可能发生以下情况： - 如果设置数据为半字(16 位)，那么数据就会传输给全部 7 个寄存器。 - 如果设置数据为字节，数据仍然会传输给全部 7 个寄存器：第一个寄存器包含第一个 MSB 节，第二个寄存器包含第一个 LSB 字节，以此类推。因此对于定时器，用户必须指定由 DMA 传输的数据宽度。
位 7:5	保留，始终读为 0。
位 4:0	**DBA[4:0]**: DMA 基地址(DMA base address) 这些位定义了 DMA 在连续模式下的基地址(当对 TIMx_DMAR 寄存器进行读或写时)，DBA 定义为从 TIMx_CR1 寄存器所在地址开始的偏移量： 00000：TIMx_CR1， 00001：TIMx_CR2， 00010：TIMx_SMCR， .....

12.4.20 TIM1 连续模式的 DMA 地址(TIMx_DMAR)

偏移地址：0x4C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0	**DMAB[15:0]**: DMA 连续传送寄存器(DMA register for burst accesses) 对 TIMx_DMAR 寄存器的读或写会导致对以下地址所在寄存器的存取操作： （TIMx_CR1 地址）+ （DBA + DMA 索引）X4， 其中： “TIMx_CR1 地址”是控制寄存器 1(TIMx_CR1)所在的地址； “DBA”是 TIMx_DCR 寄存器中定义的基地址； “DMA 索引”是由 DMA 自动控制的偏移量，它取决于 TIMx_DCR 寄存器中定义的 DBL。

12.4.21 TIM1 寄存器图

下表中将 TIM1 的所有寄存器映射到一个 16 位可寻址(编址)空间。

表 54 TIM1 – 寄存器图和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	TIMx_CR1	保留																						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
	复位值																							0	0	0	0	0	0	0	0	0	0
004h	TIMx_CR2	保留																	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	保留	CCPC
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0		0
008h	TIMx_SMCR	保留																0	0	ETPS[1:0]		EFT[3:0]				0	TS[2:0]					SMS[2:0]		
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0
00Ch	TIMx_DIER	保留																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	复位值																																
010h	TIMx_SR	保留																			0	0	0	0		0	0	0	0	0	0	0	0	0
	复位值																																
014h	TIMx_EGR	保留																								0	0	0	0	0	0	0	0
	复位值																																
018h	TIMx_CCMR1 输出比较模式	保留																0	OC2M[2:0]					CC2S[1:0]		0	OC1M[2:0]					CC1S[1:0]	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 输入捕获模式	保留																IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01Ch	TIMx_CCMR2 输出比较模式	保留																0	OC4M[2:0]					CC4S[1:0]		0	OC3M[2:0]					CC3S[1:0]	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 输入捕获模式	保留																IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	TIMx_CCER	保留																		0	0	0	0	0	0	0	0	0	0	0	0	0	0
	复位值																																
024h	TIMx_CNT	保留																CNT[15:0]															
	复 位 值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028h	TIMx_PSC	保留																PSC[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
02Ch	TIMx_ARR	保留																ARR[15:0]															
	复位值																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
030h	TIMx_RCR	保留																								REP[7:0]							
	复位值																									0	0	0	0	0	0	0	0
034h	TIMx_CCR1	保留																CCR1[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
038h	TIMx_CCR2	保留																CCR2[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03Ch	TIMx_CCR3	保留																CCR3[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
040h	TIMx_CCR4	保留																CCR4[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
044h	TIMx_BDTR	保留																MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DT[7:0]							
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
048h	TIMx_DCR	保留																			DBL[4:0]					保留	DBA[4:0]						
	复位值																				0	0	0	0	0		0	0	0	0	0	0	0
04Ch	TIMx_DMAR	保留																DMAB[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

关于寄存器的起始地址，请参见表 1。

13 通用定时器（TIMx）

13.1 TIMx 简介

通用定时器是一个通过可编程预分频器驱动的 16 位自动装载计数器构成。 它适用于多种场合，包括测量输入信号的脉冲长度(输入捕获)或者产生输出波形(输出比较和 PWM)。

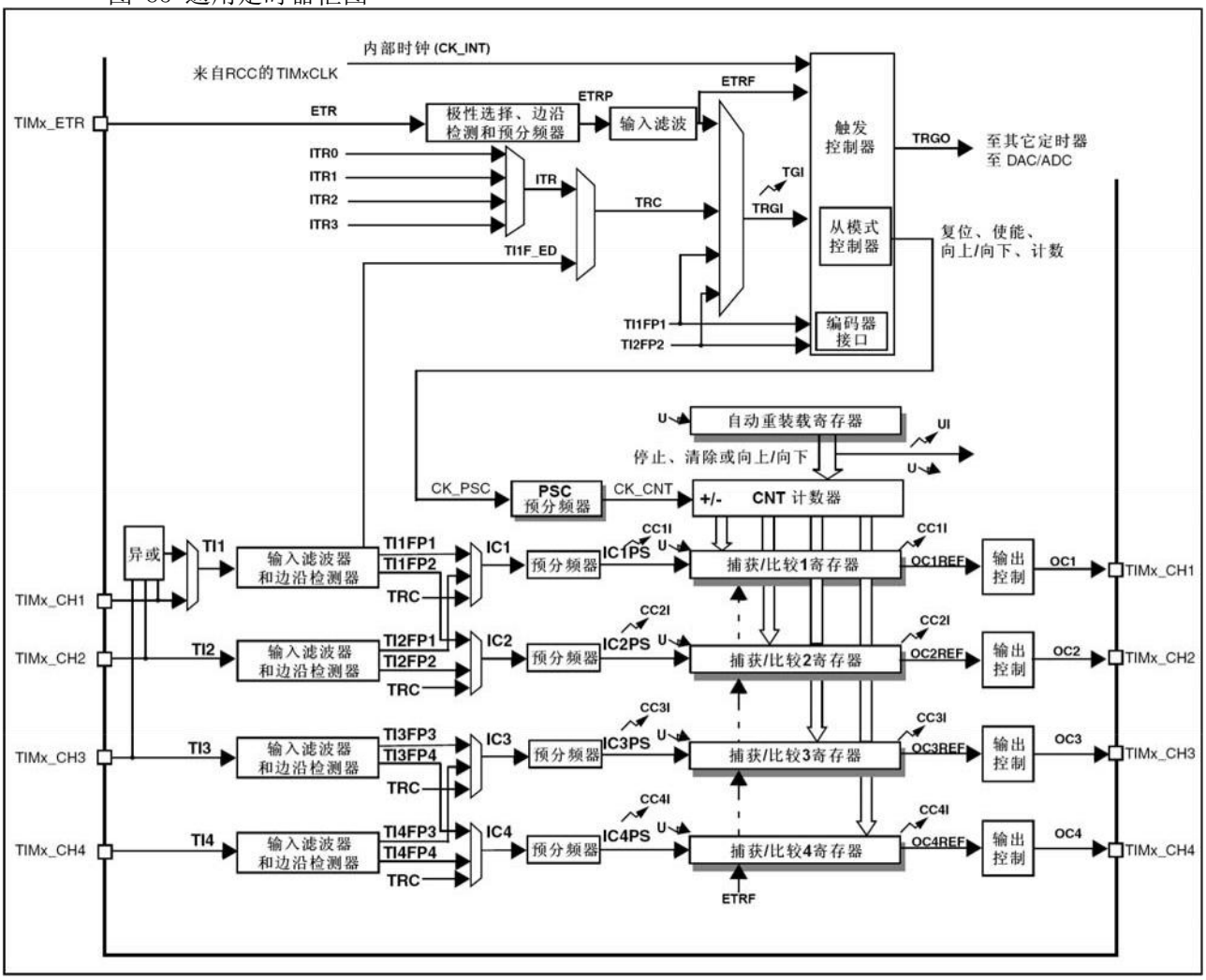
使用定时器预分频器和 RCC 时钟控制器预分频器，脉冲长度和波形周期可以在几个微秒到几个毫秒间调整。
每个定时器都是完全独立的，没有互相共享任何资源。它们可以一起同步操作，参见 14.3.15 节。

13.2 TIMx 主要功能

通用 TIMx (TIM2、TIM3、TIM4)定时器功能包括：

- 16 位向上、向下、向上/向下自动装载计数器
- 16 位可编程(可以实时修改)预分频器，计数器时钟频率的分频系数为 1～65536 之间的任意数值
- 4 个独立通道：
  - 输入捕获
  - 输出比较
  - PWM 生成(边缘或中间对齐模式)
  - 单脉冲模式输出
- 使用外部信号控制定时器和定时器互连的同步电路
- 如下事件发生时产生中断/DMA：
  - 更新：计数器向上溢出/向下溢出，计数器初始化(通过软件或者内部/外部触发)
  - 触发事件(计数器启动、停止、初始化或者由内部/外部触发计数)
  - 输入捕获
  - 输出比较
- 支持针对定位的增量(正交)编码器和霍尔传感器电路
- 触发输入作为外部时钟或者按周期的电流管理

图 85 通用定时器框图



注： *根据控制位的设定，在 U 事件时传送预加载寄存器的内容至工作寄存器*

事件

中断和 DMA 输出

13.3 TIMx 功能描述

13.3.1 时基单元

可编程通用定时器的主要部分是一个 16 位计数器和与其相关的自动装载寄存器。这个计数器可 以向上计数、向下计数或者向上向下双向计数。此计数器时钟由预分频器分频得到。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写，在计数器运行时仍可以读写。 时基单元包含：

- 计数器寄存器(TIMx_CNT)
- 预分频器寄存器(TIMx_PSC)
- 自动装载寄存器(TIMx_ARR)

自动装载寄存器是预先装载的，写或读自动重装载寄存器将访问预装载寄存器。根据在TIMx_CR1 寄存器中的自动装载预装载使能位(ARPE)的设置，预装载寄存器的内容被立即或在每次的更新事件 UEV 时传送到影子寄存器。当计数器达到溢出条件(向下计数时的下溢条件)并当TIMx_CR1 寄存器中的 UDIS 位等于’0’时，产生更新事件。更新事件也可以由软件产生。随后会详细描述每一种配置下更新事件的产生。

计数器由预分频器的时钟输出 CK_CNT 驱动，仅当设置了计数器 TIMx_CR1 寄存器中的计数器使能位(CEN)时，CK_CNT 才有效。(有关计数器使能的细节，请参见控制器的从模式描述)。

注：真正的计数器使能信号 CNT_EN 是在 CEN 的一个时钟周期后被设置。

预分频器描述

预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。它是基于一个(在 TIMx_PSC 寄存器中的)16 位寄存器控制的 16 位计数器。这个控制寄存器带有缓冲器，它能够在工作时被改变。新的预分频器参数在下一次更新事件到来时被采用。

图 86 和图 87 给出了在预分频器运行时，更改计数器参数的例子。

图 86 当预分频器的参数从 1 变到 2 时，计数器的时序图

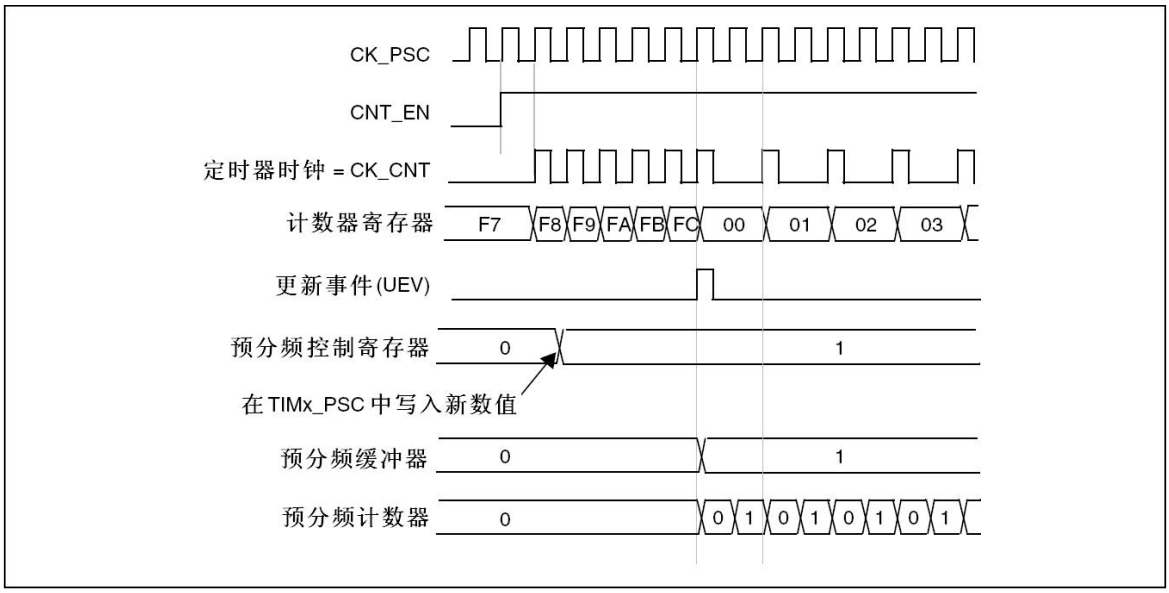
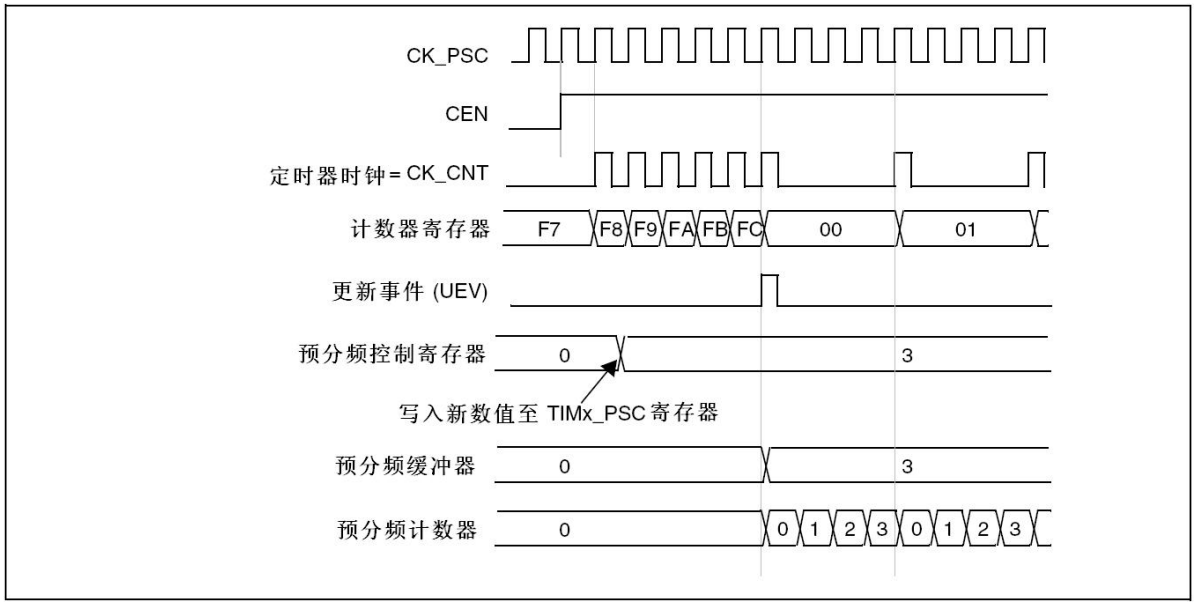


图 87 当预分频器的参数从 1 变到 4 时，计数器的时序图



13.3.2 计数器模式

向上计数模式

在向上计数模式中，计数器从 0 计数到自动加载值(TIMx_ARR 计数器的内容)，然后重新从 0 开始计数并且产生一个计数器溢出事件。

每次计数器溢出时可以产生更新事件，在 TIMx_EGR 寄存器中(通过软件方式或者使用从模式控制器)设置 UG 位也同样可以产生一个更新事件。设置 TIMx_CR1 寄存器中的 UDIS 位，可以禁止更新事件;这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清'0'之前，将不产生更新事件。但是在应该产生更新事 件时，计数器仍会被清'0'，同时预分频器的计数也被请 0(但预分频系数不变)。此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求)，设置 UG 位将产生一个更新事件 UEV，但硬件不设置 UIF 标志(即不产生中断或 DMA 请求)；这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

当发生一个更新事件时，所有的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIMx_SR 寄存器中的 UIF 位)。

- 预分频器的缓冲区被置入预装载寄存器的值(TIMx_PSC 寄存器的内容)。
- 自动装载影子寄存器被重新置入预装载寄存器的值(TIMx_ARR)。

下图给出一些例子，当 TIMx_ARR=0x36 时计数器在不同时钟频率下的动作。

图 88 计数器时序图，内部时钟分频因子为 1

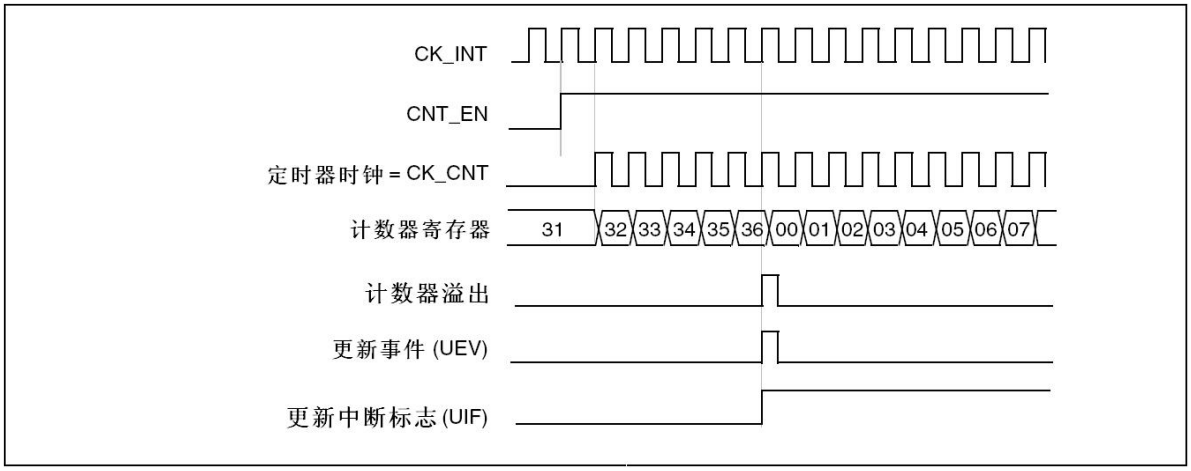


图 89 计数器时序图，内部时钟分频因子为 2

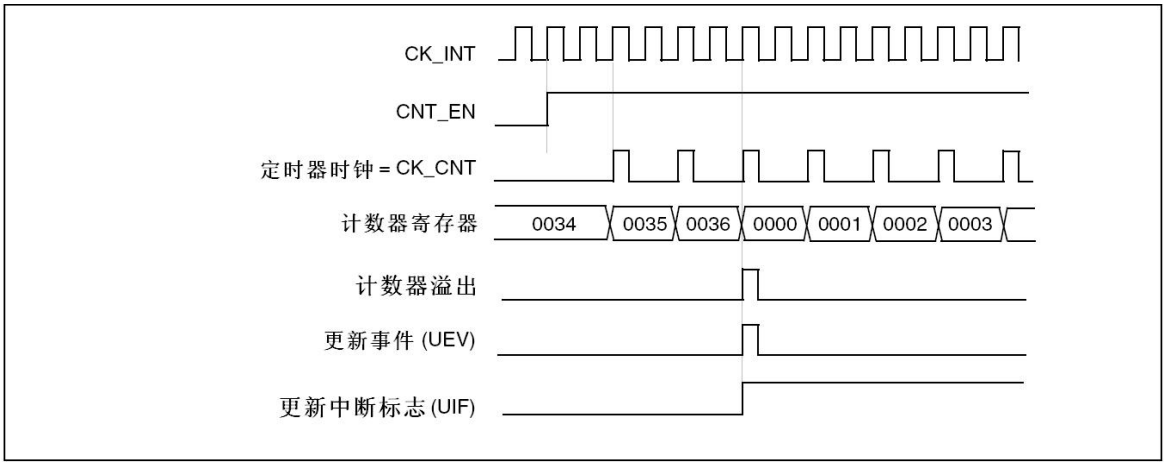


图 90 计数器时序图，内部时钟分频因子为 4

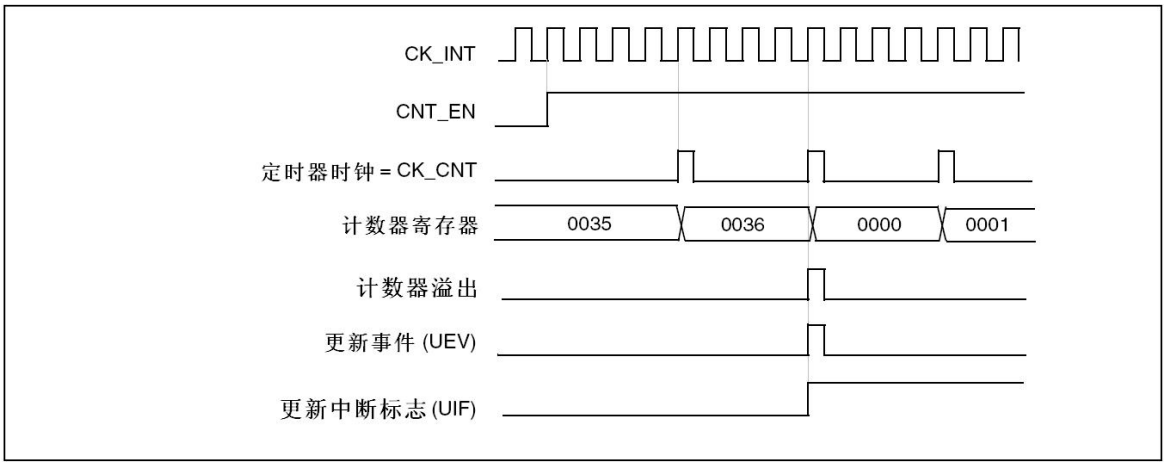


图 91 计数器时序图，内部时钟分频因子为 N

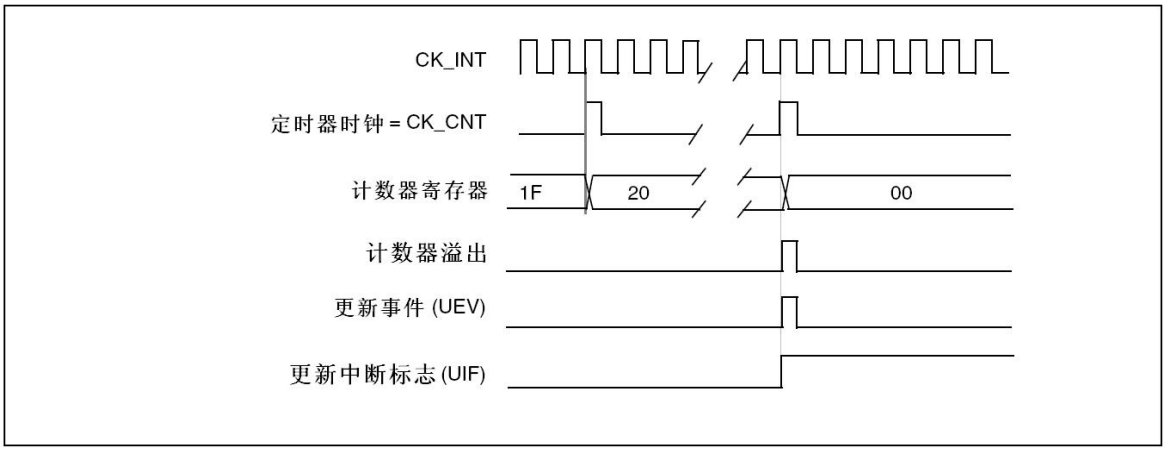


图 92 计数器时序图，当 ARPE=0 时的更新事件(TIMx_ARR 没有预装入)

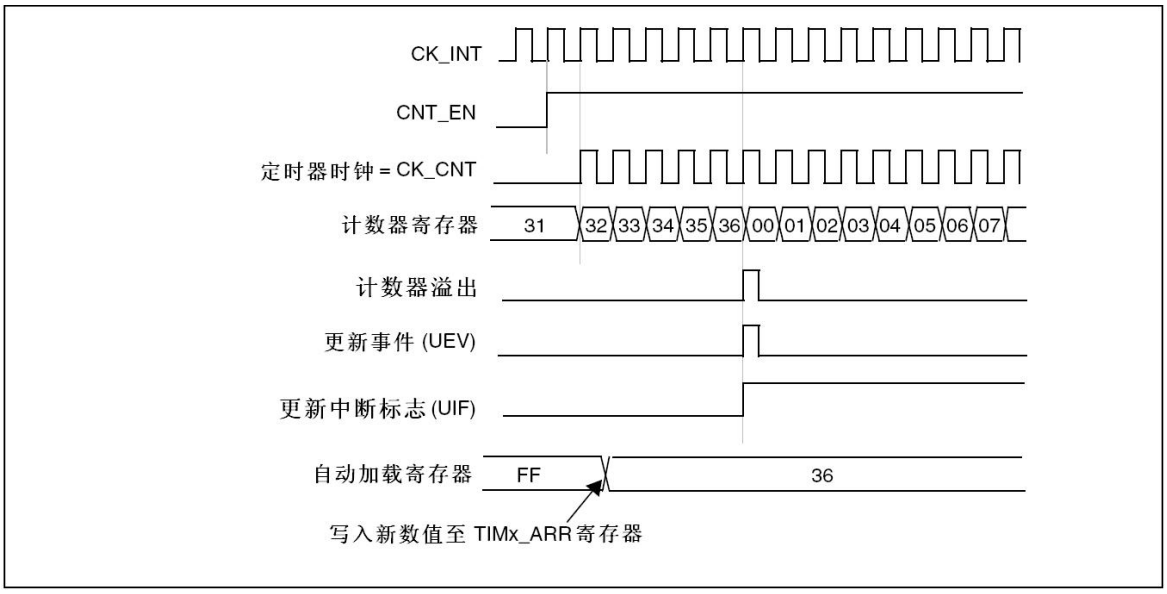
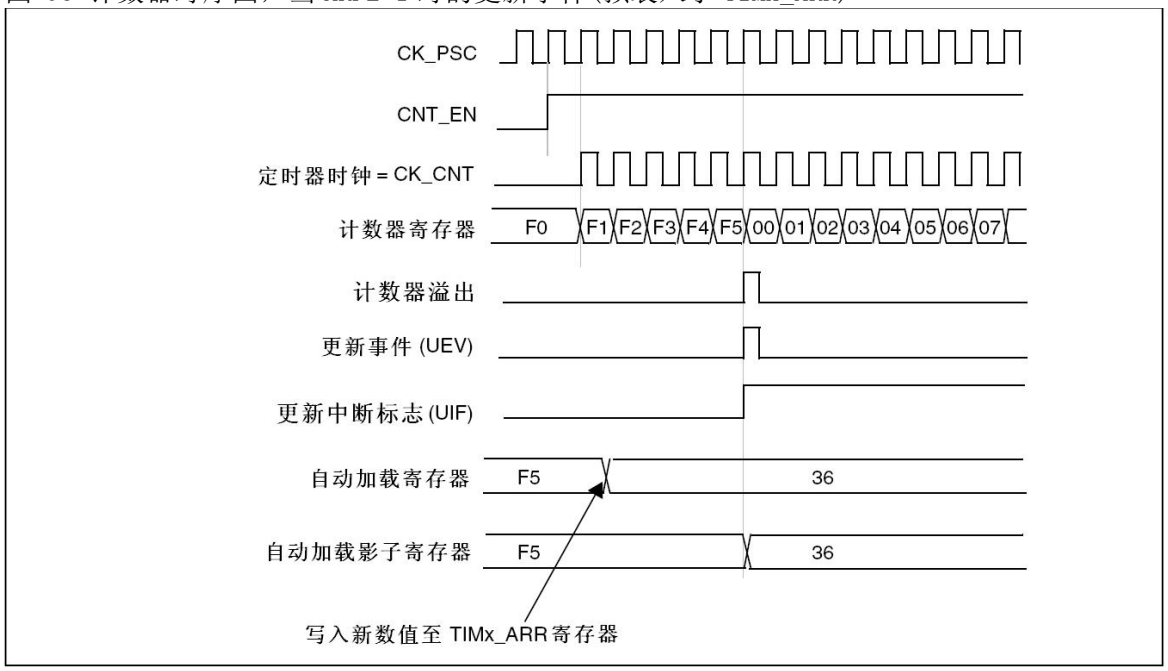


图 93 计数器时序图，当 ARPE=1 时的更新事件(预装入了 TIMx_ARR)



向下计数模式

在向下模式中，计数器从自动装入的值(TIMx_ARR 计数器的值)开始向下计数到 0，然后从自动装入的值重新开始并且产生一个计数器向下溢出事件。

每次计数器溢出时可以产生更新事件，在 TIMx_EGR 寄存器中(通过软件方式或者使用从模式控制器)设置 UG 位，也同样可以产生一个更新事件。

设置 TIMx_CR1 寄存器的 UDIS 位可以禁止 UEV 事件。这样可以避免向预装载寄存器中写入新值时更新影子寄存器。因此 UDIS 位被清为'0'之前不会产生更新事件。然而，计数器仍会从当前自 动加载值重新开始计数，同时预分频器的计数器重新从 0 开始(但预分频系数不变)。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求) ，设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志(因此不产生中断和 DMA 请求)，这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且(根据 URS 位的设置)更新标志位(TIMx_SR 寄存器中的 UIF 位)也被设置。

- 预分频器的缓存器被置入预装载寄存器的值(TIMx_PSC 寄存器的值)。
- 当前的自动加载寄存器被更新为预装载值(TIMx_ARR 寄存器中的内容)。

注：自动装载在计 数器重载入之前被更新，因此下一个周期将是预期的值。

以下是一些当 TIMx_ARR=0x36 时，计数器在不同时钟频率下的操作例子。

图 94 计数器时序图，内部时钟分频因子为 1

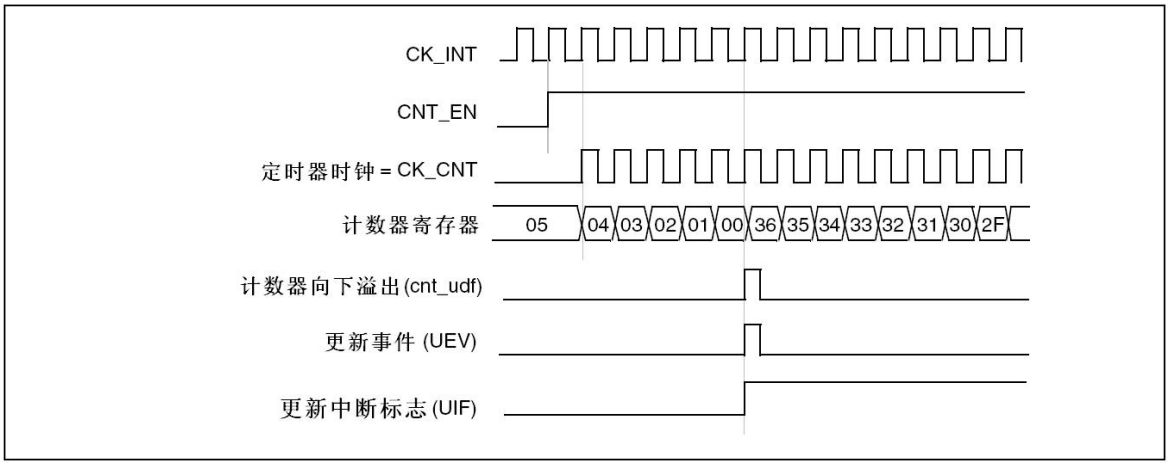


图 95 计数器时序图，内部时钟分频因子为 2

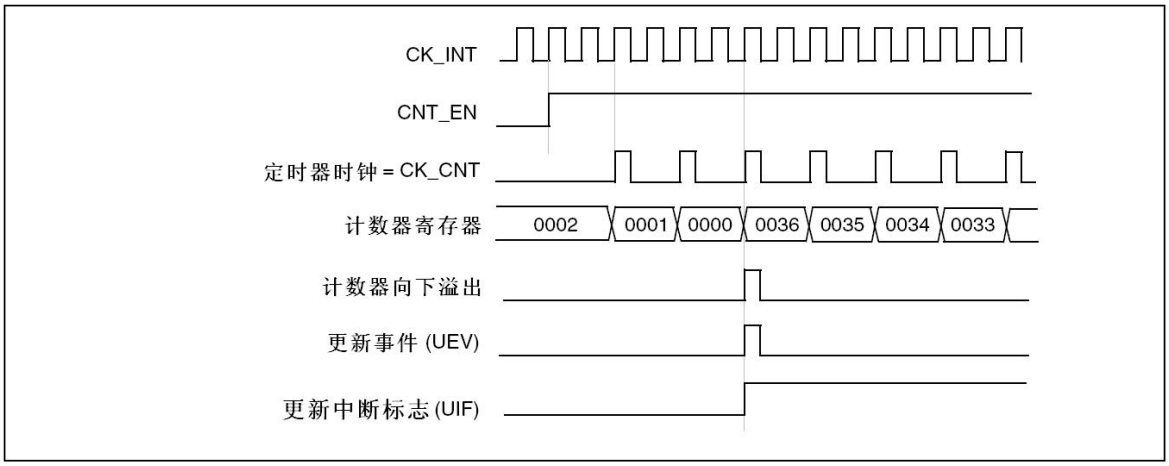


图 96 计数器时序图，内部时钟分频因子为 4

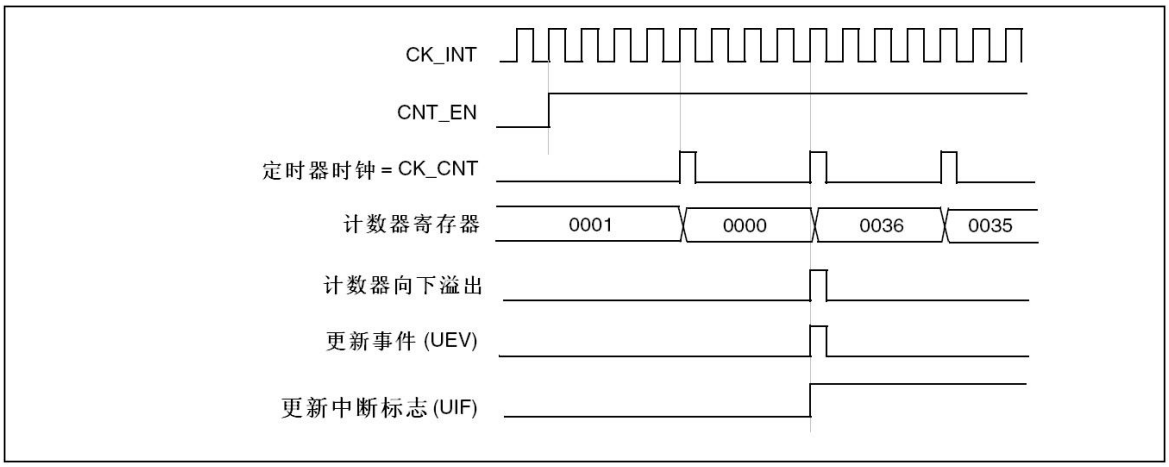


图 97 计数器时序图，内部时钟分频因子为 N

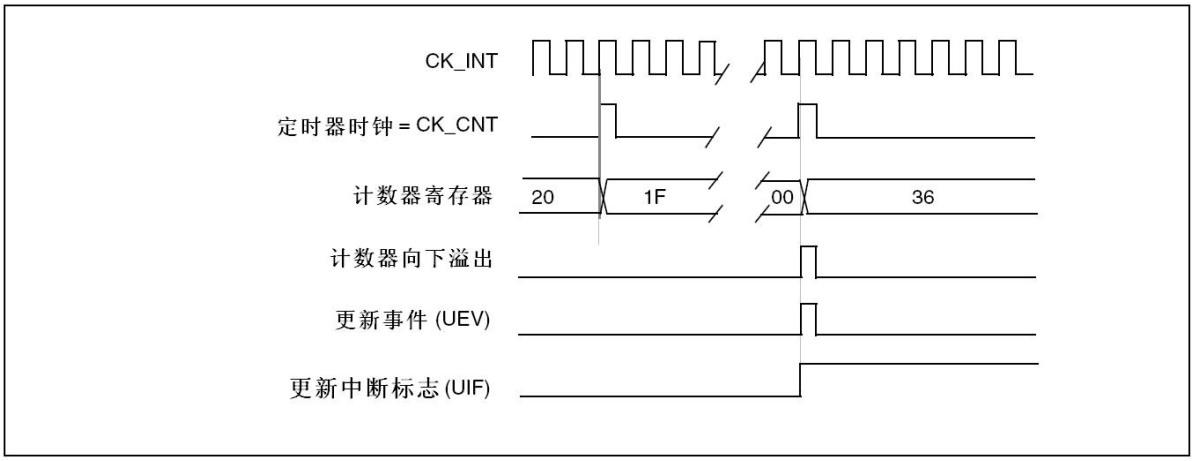
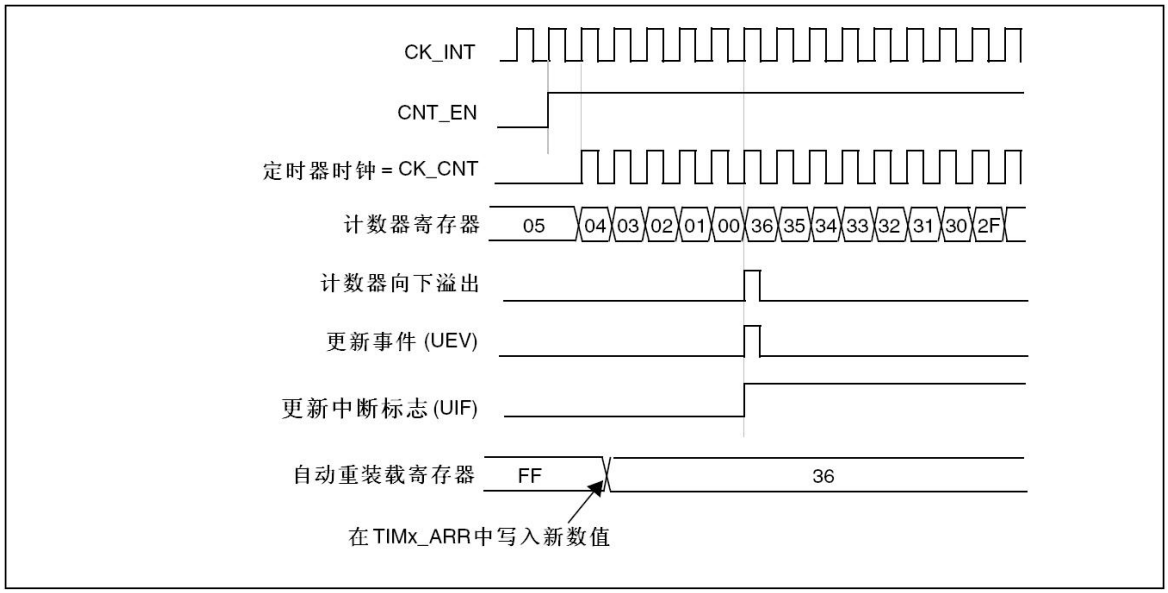


图 98 计数器时序图，当没有使用重复计数器时的更新事件



中央对齐模式(向上/向下计数)

在中央对齐模式，计数器从 0 开始计数到自动加载的值(TIMx_ARR 寄存器)−1，产生一个计数器溢出事件，然后向下计数到 1 并且产生一个计数器下溢事件；然后再从 0 开始重新计数。

在这个模式，不能写入 TIMx_CR1 中的 DIR 方向位。它由硬件更新并指示当前的计数方向。

可以在每次计数上溢和每次计数下溢时产生更新事件；也可以通过(软件或者使用从模式控制器)设置 TIMx_EGR 寄存器中的 UG 位产生更新事件。然后，计数器重新从 0 开始计数，预分频器也重新从 0 开始计数。

设置 TIMx_CR1 寄存器中的 UDIS 位可以禁止 UEV 事件。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。因此 UDIS 位被清为'0'之前不会产生更新事件。然而，计数器仍会根据 当前自动重加载的值，继续向上或向下计数。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求) ，设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志(因此不产生中断和 DMA 请求)，这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且(根据 URS 位的设置)更新标志位(TIMx_SR 寄存器中的 UIF 位)也被设置。

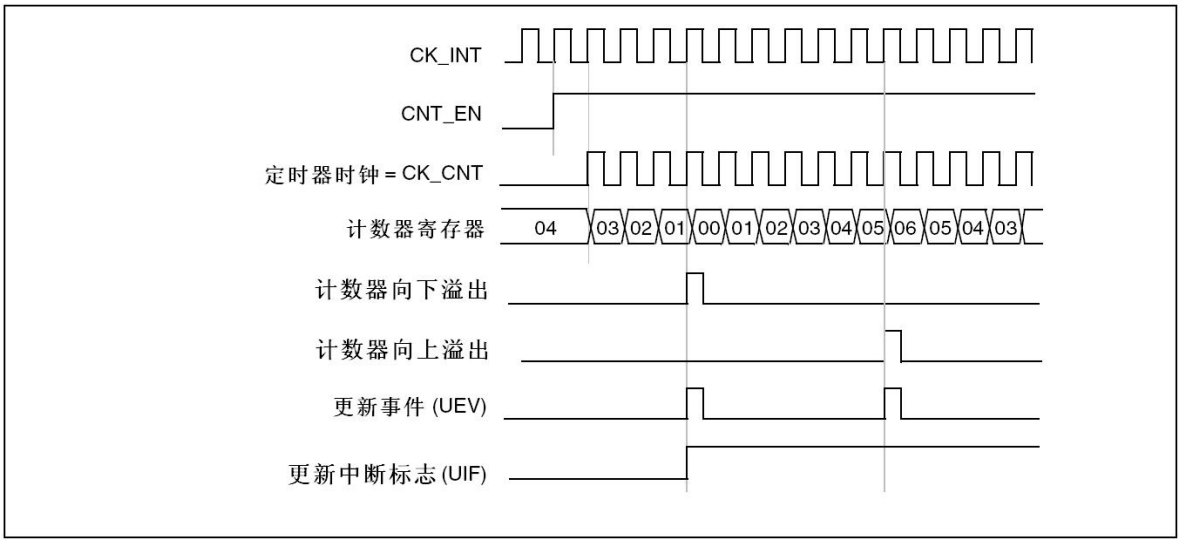
- 预分频器的缓存器被加载为预装载(TIMx_PSC 寄存器)的值。
- 当前的自动加载寄存器被更新为预装载值(TIMx_ARR 寄存器中的内容)。

注：如果因为计数器溢出而产生更新，自动重装载将在计数器重载入之前被更新，因此下一个周

期将是预期的值(计数器被装载为新的值)。

以下是一些计数器在不同时钟频率下的操作的例子：

图 99 计数器时序图，内部时钟分频因子为 1，TIMx_ARR=0x6



1．这里使用了中心对齐模式 1(详见 14.4.1 节)。

图 100 计数器时序图，内部时钟分频因子为 2

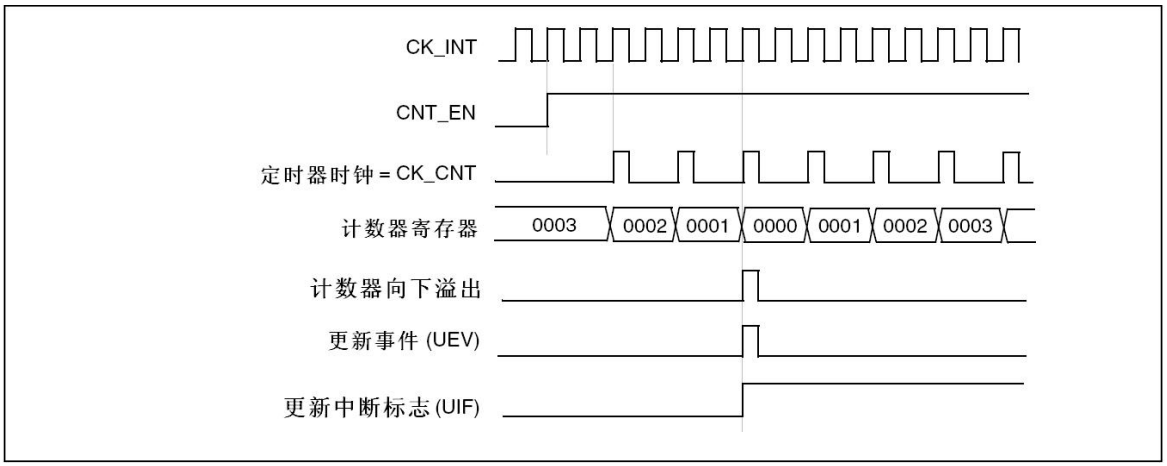


图 101 计数器时序图，内部时钟分频因子为 4，TIMx_ARR=0x36

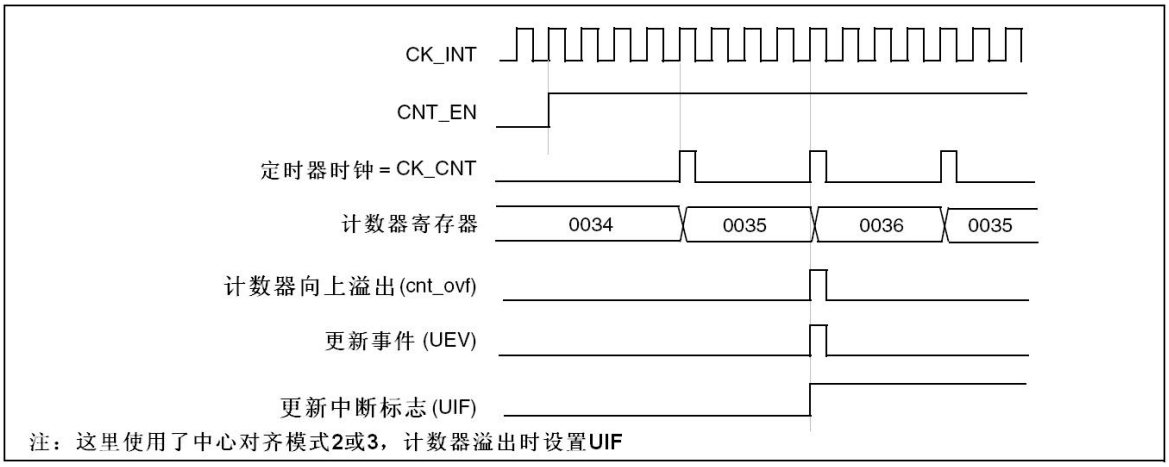


图 102 计数器时序图，内部时钟分频因子为 N

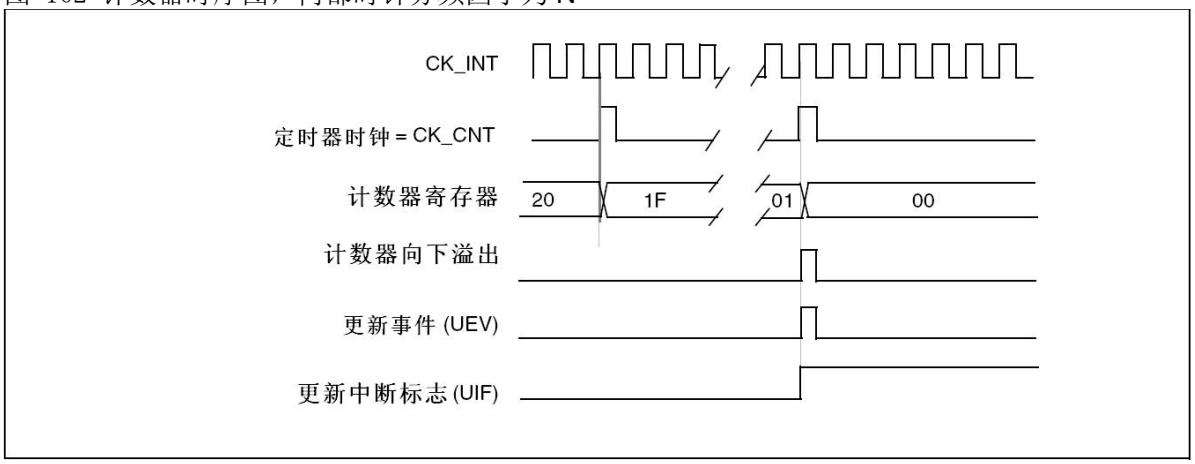


图 103 计数器时序图，ARPE=1 时的更新事件(计数器下溢)

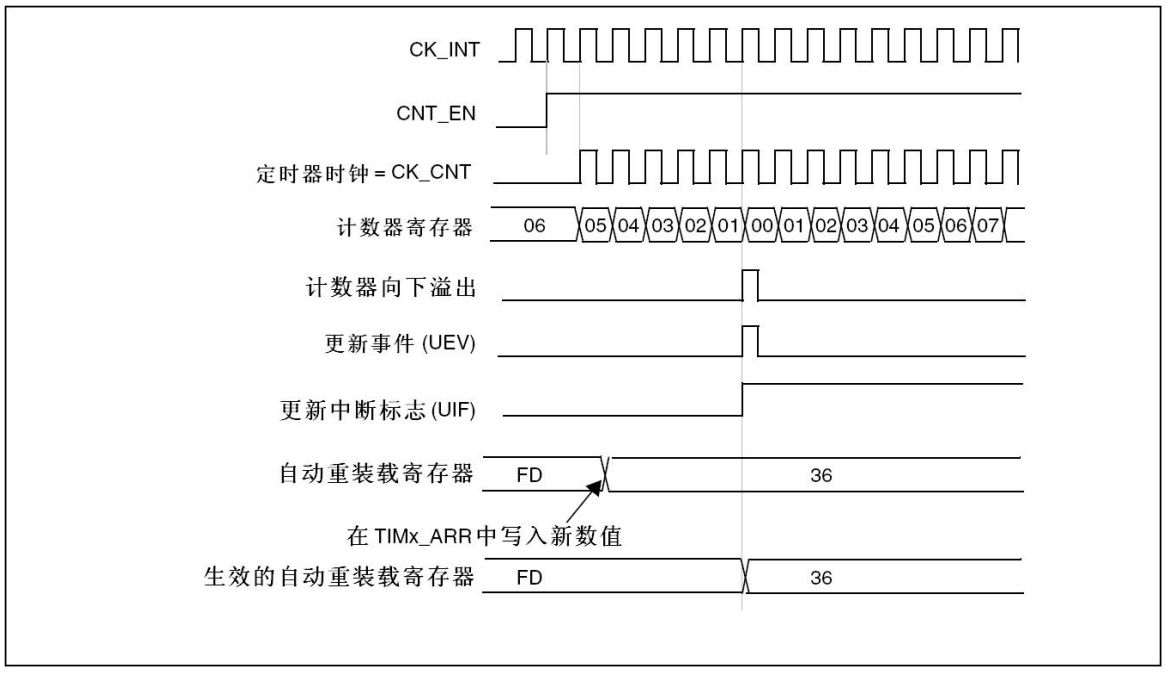
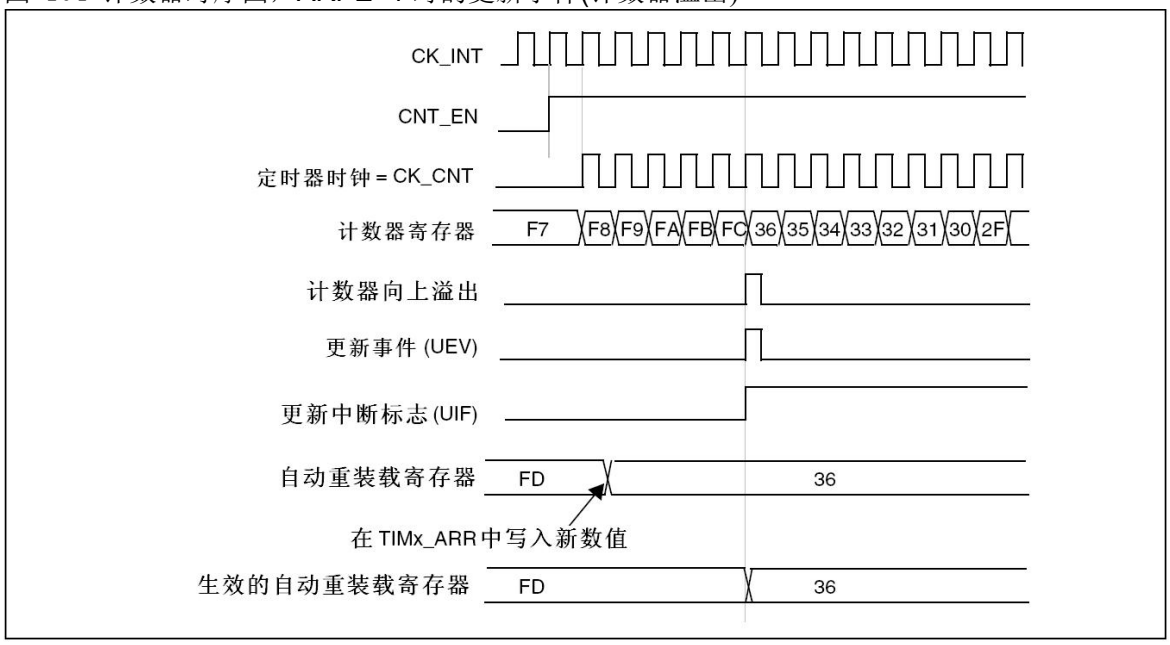


图 104 计数器时序图，ARPE=1 时的更新事件(计数器溢出)



13.3.3 时钟选择

计数器时钟可由下列时钟源提供：

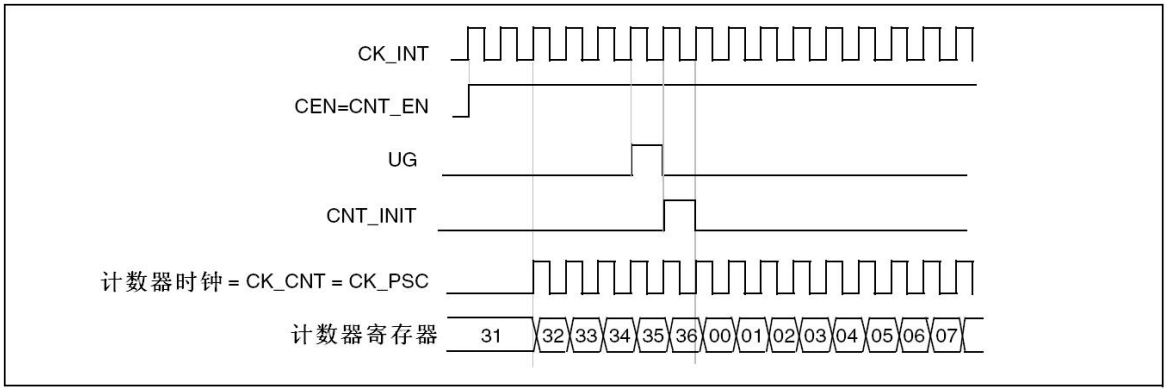
- 内部时钟(CK_INT)
- 外部时钟模式 1：外部输入脚(TIx)
- 外部时钟模式 2：外部触发输入(ETR)
- 内部触发输入(ITRx)：使用一个定时器作为另一个定时器的预分频器，如可以配置一个定时器 Timer1 而作为另一个定时器 Timer2 的预分频器。参见 14.3.15。

内部时钟源(CK_INT)

如果禁止了从模式控制器(TIMx_SMCR 寄存器的 SMS=000)，则 CEN、DIR(TIMx_CR1 寄存器)和 UG 位(TIMx_EGR 寄存器)是事实上的控制位，并且只能被软件修改(UG 位仍被自动清除)。只要 CEN 位被写成'1'，预分频器的时钟就由内部时钟 CK_INT 提供。

下图显示了控制电路和向上计数器在一般模式下，不带预分频器时的操作。

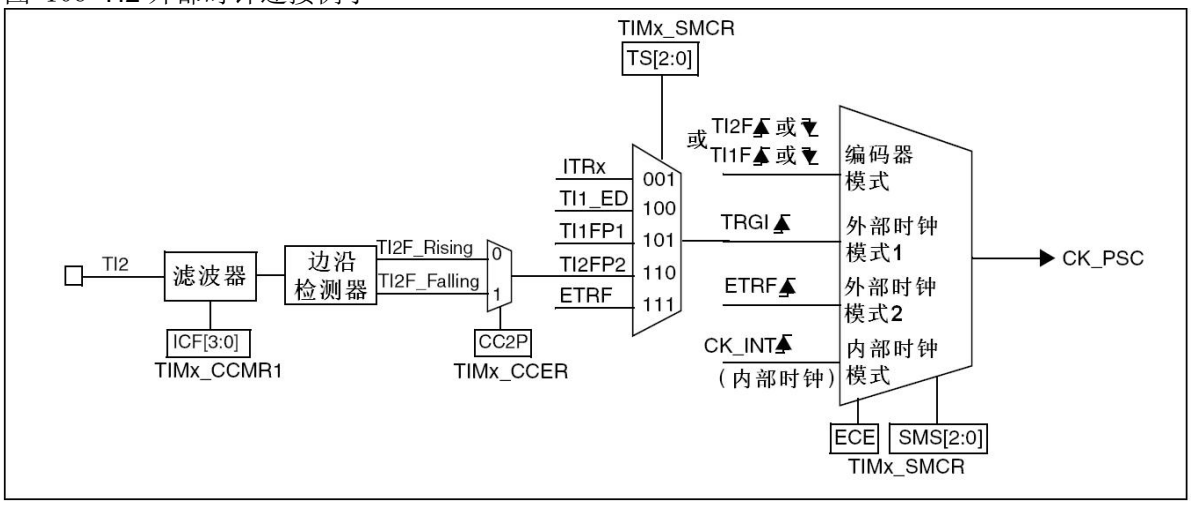
图 105 一般模式下的控制电路，内部时钟分频因子为 1



外部时钟源模式 1

当 TIMx_SMCR 寄存器的 SMS=111 时，此模式被选中。计数器可以在选定输入端的每个上升沿或下降沿计数。

图 106 TI2 外部时钟连接例子



例如，要配置向上计数器在 T12 输入端的上升沿计数，使用下列步骤：

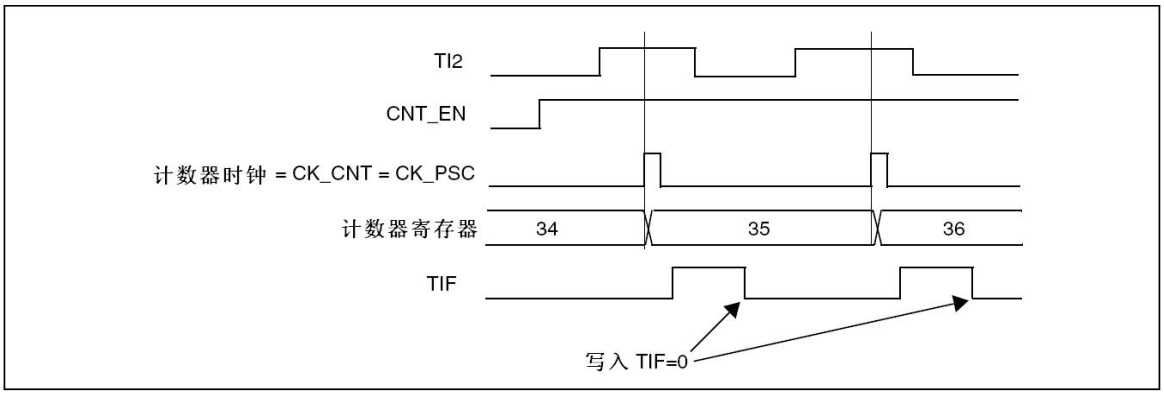
1. 配置 TIMx_CCMR1 寄存器 CC2S='01'，配置通道 2 检测 TI2 输入的上升沿
2. 配置 TIMx_CCMR1 寄存器的 IC2F[3:0]，选择输入滤波器带宽(如果不需要滤波器，保持 IC2F=0000)

注：　捕获预分频器不用作触发，所以不需要对它进行配置

3. 配置 TIMx_CCER 寄存器的 CC2P='0'，选定上升沿极性
4. 配置 TIMx_SMCR 寄存器的 SMS='111'，选择定时器外部时钟模式 1
5. 配置 TIMx_SMCR 寄存器中的 TS='110'，选定 TI2 作为触发输入源
6. 设置 TIMx_CR1 寄存器的 CEN='1'，启动计数器

当上升沿出现在 TI2，计数器计数一次，且 TIF 标志被设置。 在 TI2 的上升沿和计数器实际时钟之间的延时，取决于在 TI2 输入端的重新同步电路。

图 107 外部时钟模式 1 下的控制电路



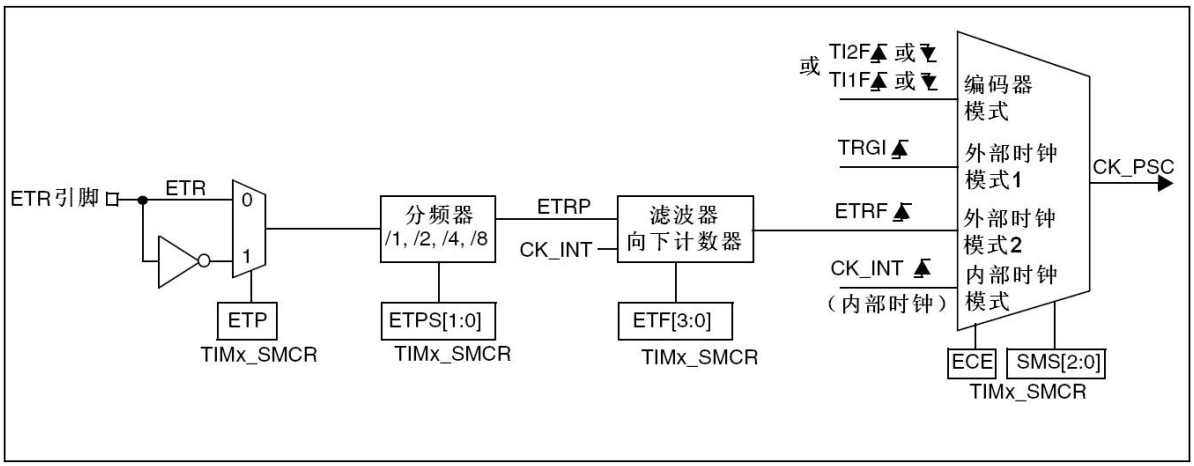
外部时钟源模式 2

选定此模式的方法为：令 TIMx_SMCR 寄存器中的 ECE=1

计数器能够在外部触发 ETR 的每一个上升沿或下降沿计数。

下图是外部触发输入的框图

图 108 外部触发输入框图



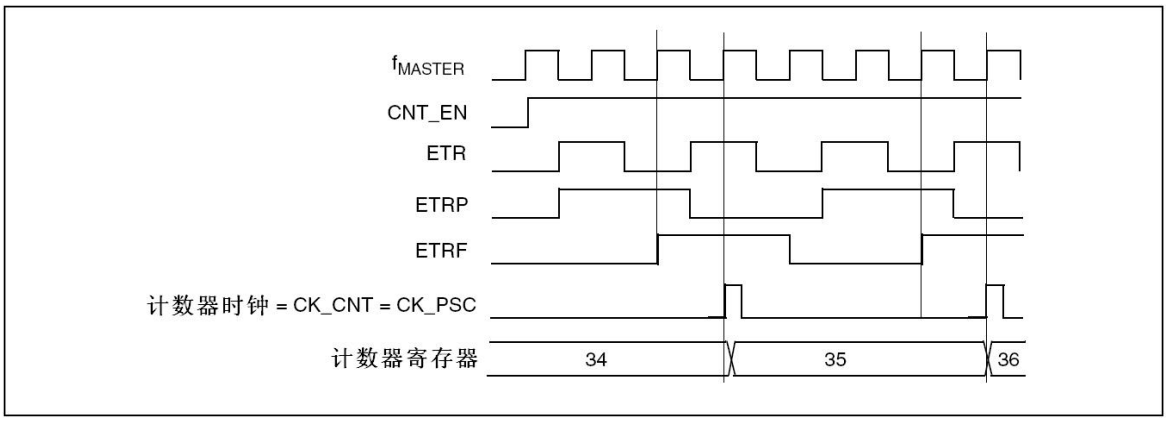
例如，要配置在 ETR 下每 2 个上升沿计数一次的向上计数器，使用下列步骤：

1. 本例中不需要滤波器，置 TIMx_SMCR 寄存器中的 ETF[3:0]=0000
2. 设置预分频器，置 TIMx_SMCR 寄存器中的 ETPS[1:0]=01
3. 设置在 ETR 的上升沿检测，置 TIMx_SMCR 寄存器中的 ETP=0
4. 开启外部时钟模式 2，置 TIMx_SMCR 寄存器中的 ECE=1
5. 启动计数器，置 TIMx_CR1 寄存器中的 CEN=1

计数器在每 2 个 ETR 上升沿计数一次。

在 ETR 的上升沿和计数器实际时钟之间的延时取决于在 ETRP 信号端的重新同步电路。

图 109 外部时钟模式 2 下的控制电路



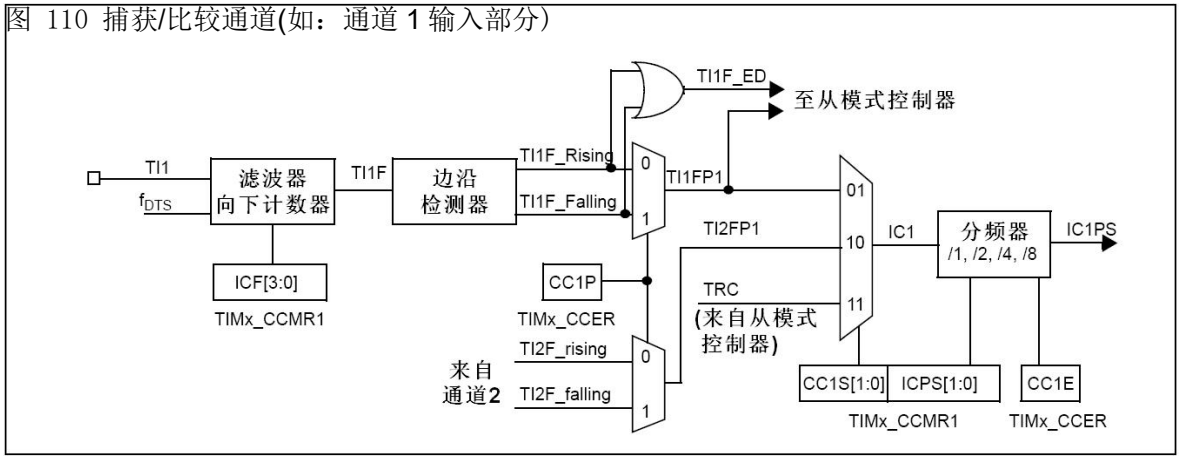
13.3.4 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器(包含影子寄存器)，包括捕获的输入部分(数字滤波、多路复用和预分频器)，和输出部分(比较器和输出控制)。

下面几张图是一个捕获/比较通道概览。

输入部分对相应的 TIx 输入信号采样，并产生一个滤波后的信号 TIxF。然后，一个带极性选择的边缘检测器产生一个信号(TIxFPx)，它可以作为从模式控制器的输入触发或者作为捕获控制。该信号通过预分频进入捕获寄存器(ICxPS)。

图 110 捕获/比较通道(如：通道 1 输入部分)



输出部分产生一个中间波形 OCxRef(高有效)作为基准，链的末端决定最终输出信号的极性。

图 111 捕获/比较通道 1 的主电路

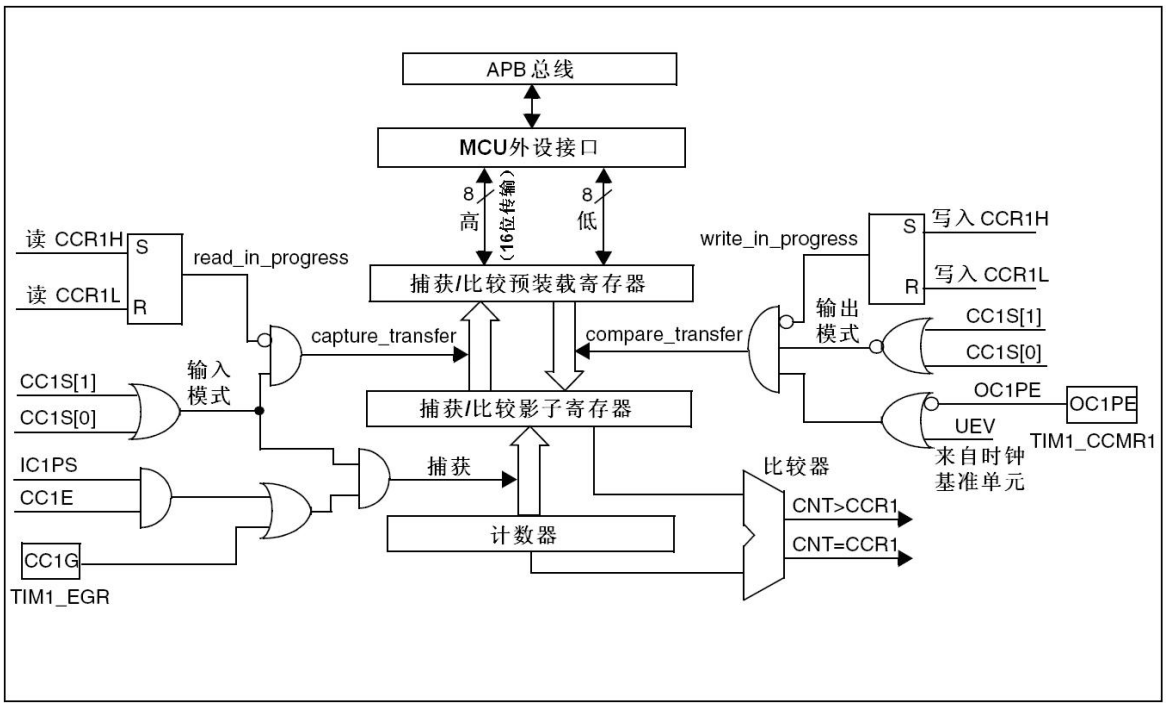
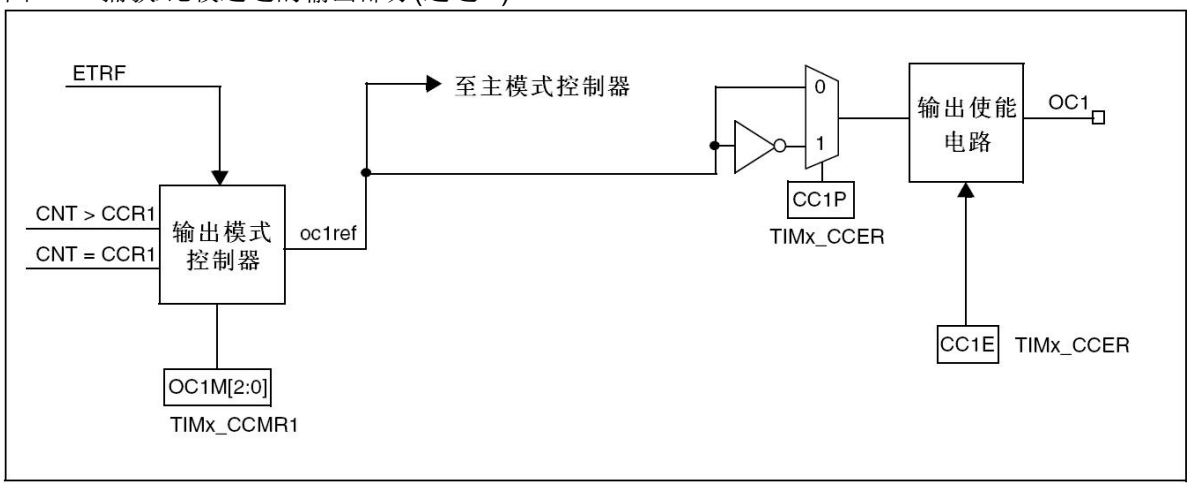


图 112 捕获/比较通道的输出部分(通道 1)



捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。
在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。
在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后影子寄存器的内容和计数器进行比较。

13.3.5 输入捕获模式

在输入捕获模式下，当检测到 ICx 信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器(TIMx_CCRx)中。当捕获事件发生时，相应的 CCxIF 标志(TIMx_SR 寄存器)被置'1'，如果使能 了中断或者 DMA 操作，则将产生中断或者 DMA 操作。如果捕获事件发生时 CCxIF 标志已经为高，那么重复捕获标志 CCxOF(TIMx_SR 寄存器)被置'1'。写 CCxIF=0 可清除 CCxIF，或读取存 储在 TIMx_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0 可清除 CCxOF。
以下例子说明如何在 TI1 输入的上升沿时捕获计数器的值到 TIMx_CCR1 寄存器中，步骤如下：

- 选择有效输入端：TIMx_CCR1 必须连接到 TI1 输入，所以写入 TIMx_CCR1 寄存器中的 CC1S=01，只要 CC1S 不为'00'，通道被配置为输入，并且 TM1_CCR1 寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽(即输入为 TIx 时，输入滤波器控制位是 TIMx_CCMRx 寄存器中的 ICxF 位)。假设输入信号在最多 5 个内部时钟周期的时间内抖动，我们须配置滤波器的带宽长于 5 个时钟周期。因此我们可以(以 f_{DTS} 频率)连续采样 8 次，以确认在 TI1 上一次真实的边沿变换，即在 TIMx_CCMR1 寄存器中写入 IC1F=0011。
- 选择 TI1 通道的有效转换边沿，在 TIMx_CCER 寄存器中写入 CC1P=0(上升沿)。
- 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止(写 TIMx_CCMR1 寄存器的 IC1PS=00)。
- 设置 TIMx_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，通过设置 TIMx_DIER 寄存器中的 CC1IE 位允许相关中断请求，通过设置 TIMx_DIER 寄存器中的 CC1DE 位允许 DMA 请求。

当发生一个输入捕获时：

- 产生有效的电平转换时，计数器的值被传送到 TIMx_CCR1 寄存器。
- CC1IF 标志被设置(中断标志)。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除，CC1OF 也被置'1'。
- 如设置了 CC1IE 位，则会产生一个中断。
- 如设置了 CC1DE 位，则还会产生一个 DMA 请求。

为了处理捕获溢出，建议在读出捕获溢出标志之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注：设置 TIMx_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断和/或 DMA 请求。

13.3.6 PWM 输入模式

该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

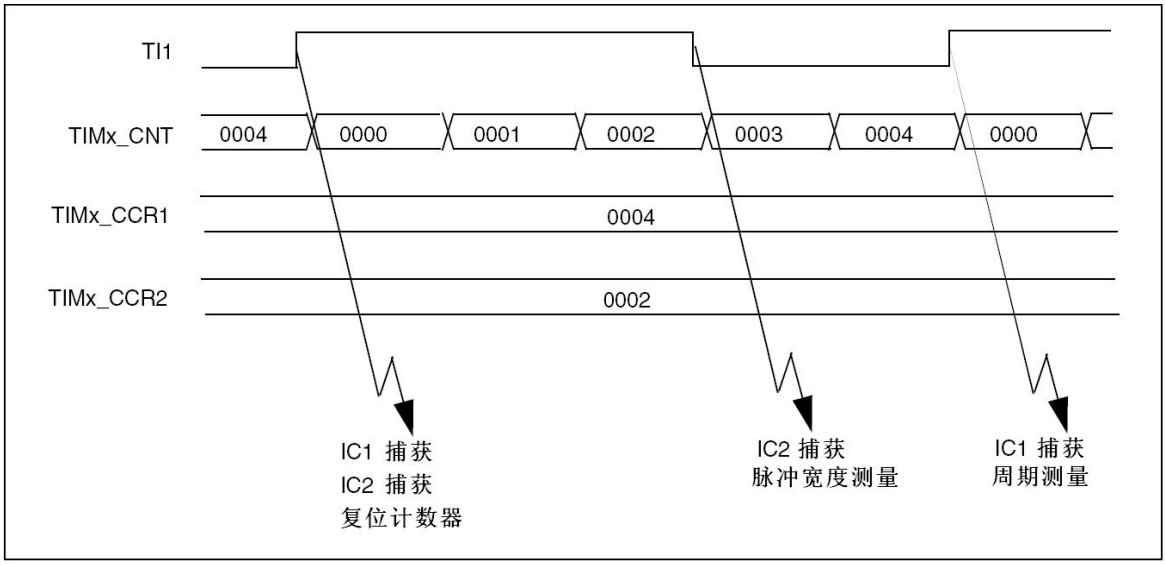
- 两个 ICx 信号被映射至同一个 TIx 输入。
- 这 2 个 ICx 信号为边沿有效，但是极性相反。
- 其中一个 TIxFP 信号被作为触发输入信号，而从模式控制器被配置成复位模式。

例如，你需要测量输入到 TI1 上的 PWM 信号的长度(TIMx_CCR1 寄存器)和占空比(TIMx_CCR2 寄存器)，具体步骤如下(取决于 CK_INT 的频率和预分频器的值)

- 选择 TIMx_CCR1 的有效输入：置 TIMx_CCMR1 寄存器的 CC1S=01(选择 TI1)。
- 选择 TI1FP1 的有效极性(用来捕获数据到 TIMx_CCR1 中和清除计数器)：置 CC1P=0(上升沿有效)。
- 选择 TIMx_CCR2 的有效输入：置 TIMx_CCMR1 寄存器的 CC2S=10(选择 TI1)。
- 选择 TI1FP2 的有效极性(捕获数据到 TIMx_CCR2)：置 CC2P=1(下降沿有效)。
- 选择有效的触发输入信号：置 TIMx_SMCR 寄存器中的 TS=101(选择 TI1FP1)。

- 配置从模式控制器为复位模式：置 TIMx_SMCR 中的 SMS=100。
- 使能捕获：置 TIMx_CCER 寄存器中 CC1E=1 且 CC2E=1。

图 113 PWM 输入模式时序



由于只有 TI1FP1 和 TI2FP2 连到了从模式控制器，所以 PWM 输入模式只能使用 TIMx_CH1/TIMx_CH2 信号。

13.3.7 强置输出模式

在输出模式(TIMx_CCMRx 寄存器中 CCxS=00)下，输出比较信号(OCxREF 和相应的 OCx)能够直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。

置 TIMx_CCMRx 寄存器中相应的 OCxM=101，即可强置输出比较信号(OCxREF/OCx)为有效状态。这样 OCxREF 被强置为高电平(OCxREF 始终为高电平有效)，同时 OCx 得到 CCxP 极性位相反的值。

例如：CCxP=0(OCx 高电平有效)，则 OCx 被强置为高电平。

置 TIMx_CCMRx 寄存器中的 OCxM=100，可强置 OCxREF 信号为低。

该模式下，在 TIMx_CCRx 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。因此仍然会产生相应的中断和 DMA 请求。这将会在下面的输出比较模式一节中介绍。

13.3.8 输出比较模式

此项功能是用来控制一个输出波形，或者指示一段给定的的时间已经到时。 当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作：

- 将输出比较模式(TIMx_CCMRx 寄存器中的 OCxM 位)和输出极性(TIMx_CCER 寄存器中的 CCxP 位)定义的值输出到对应的引脚上。在比较匹配时，输出引脚可以保持它的电平(OCxM=000)、被设置成有效电平(OCxM=001)、被设置成无效电平(OCxM=010)或进行翻 转(OCxM=011)。
- 设置中断状态寄存器中的标志位(TIMx_SR 寄存器中的 CCxIF 位)。
- 若设置了相应的中断屏蔽(TIMx_DIER 寄存器中的 CCxIE 位)，则产生一个中断。
- 若设置了相应的使能位(TIMx_DIER 寄存器中的 CCxDE 位，TIMx_CR2 寄存器中的 CCDS 位 选择 DMA 请求功能)，则产生一个 DMA 请求。

TIMx_CCMRx 中的 OCxPE 位选择 TIMx_CCRx 寄存器是否需要使用预装载寄存器。

在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。

同步的精度可以达到计数器的一个计数周期。输出比较模式(在单脉冲模式下)也能用来输出一个

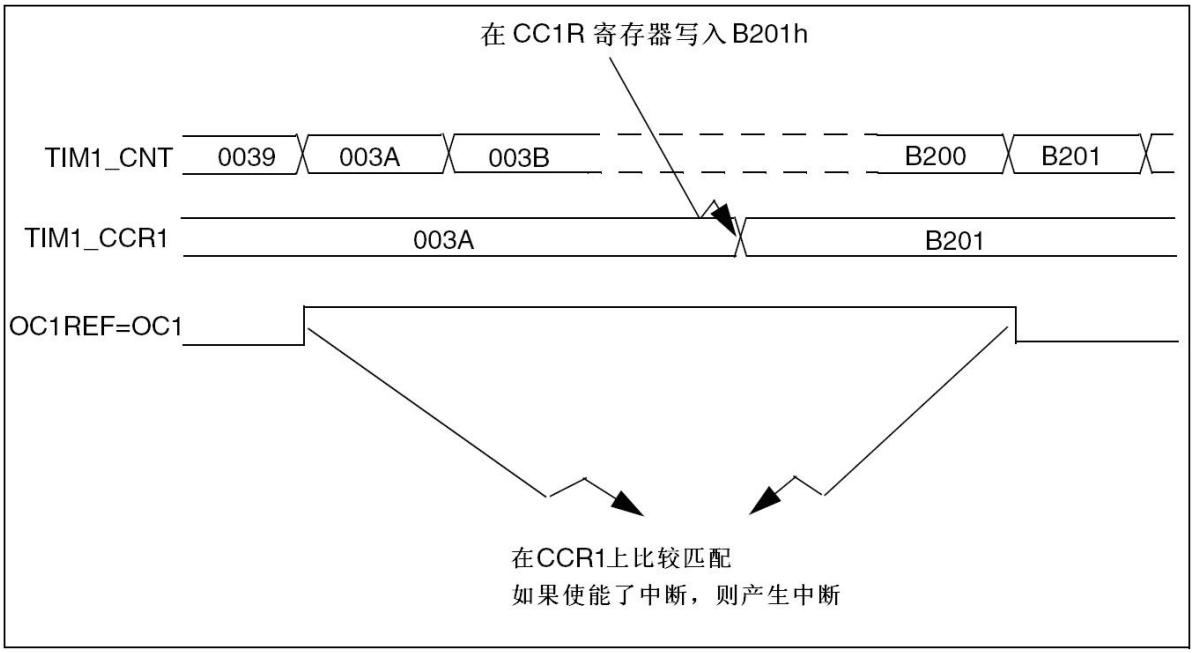
单脉冲。

输出比较模式的配置步骤：

1. 选择计数器时钟(内部，外部，预分频器)
2. 将相应的数据写入 TIMx_ARR 和 TIMx_CCRx 寄存器中
3. 如果要产生一个中断请求和/或一个 DMA 请求，设置 CCxIE 位和/或 CCxDE 位。
4. 选择输出模式，例如当计数器 CNT 与 CCRx 匹配时翻转 OCx 的输出引脚，CCRx 预装载未 用，开启 OCx 输出且高电平有效，则必须设置 OCxM=’011’、OCxPE=’0’、CCxP=’0’和 CCxE=’1’。
5. 设置 TIMx_CR1 寄存器的 CEN 位启动计数器

TIMx_CCRx 寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄存器(OCxPE=’0’，否则 TIMx_CCRx 影子寄存器只能在发生下一次更新事件时被更新)。下图给出了一个例子。

图 114 输出比较模式，翻转 OC1



13.3.9 PWM 模式

脉冲宽度调制模式可以产生一个由 TIMx_ARR 寄存器确定频率、由 TIMx_CCRx 寄存器确定占空比的信号。

在 TIMx_CCMRx 寄存器中的 OCxM 位写入’110’(PWM 模式 1)或’111’(PWM 模式 2)，能够独立地设置每个 OCx 输出通道产生一路 PWM。必须设置 TIMx_CCMRx 寄存器 OCxPE 位以使能相应的预装载寄存器，最后还要设置 TIMx_CR1 寄存器的 ARPE 位，(在向上计数或中心对称模式中)使能自动重装载的预装载寄存器。

仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，必须通过设置 TIMx_EGR 寄存器中的 UG 位来初始化所有的寄存器。 OCx 的极性可以通过软件在 TIMx_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或低电平有效。TIMx_CCER 寄存器中的 CCxE 位控制 OCx 输出使能。详见 TIMx_CCERx 寄存器的描述。

在 PWM 模式(模式 1 或模式 2)下，TIMx_CNT 和 TIMx_CCRx 始终在进行比较，(依据计数器的计数 方向) 以确定是否符合 TIMx_CCRx≤TIMx_CNT 或者 TIMx_CNT≤TIMx_CCRx 。然而为了与 OCREF_CLR 的功能(在下一个 PWM 周期之前，ETR 信号上的一个外部事件能够清除 OCxREF)一致，OCxREF 信号只能在下述条件下产生：

- 当比较的结果改变，或

- 当输出比较模式(TIMx_CCMRx 寄存器中的 OCxM 位)从“冻结”(无比较，OCxM=’000’)切 换到某个 PWM 模式(OCxM=’110’或’111’)。

这样在运行中可以通过软件强置 PWM 输出。 根据 TIMx_CR1 寄存器中 CMS 位的状态，定时器能够产生边沿对齐的 PWM 信号或中央对齐的 PWM 信号。

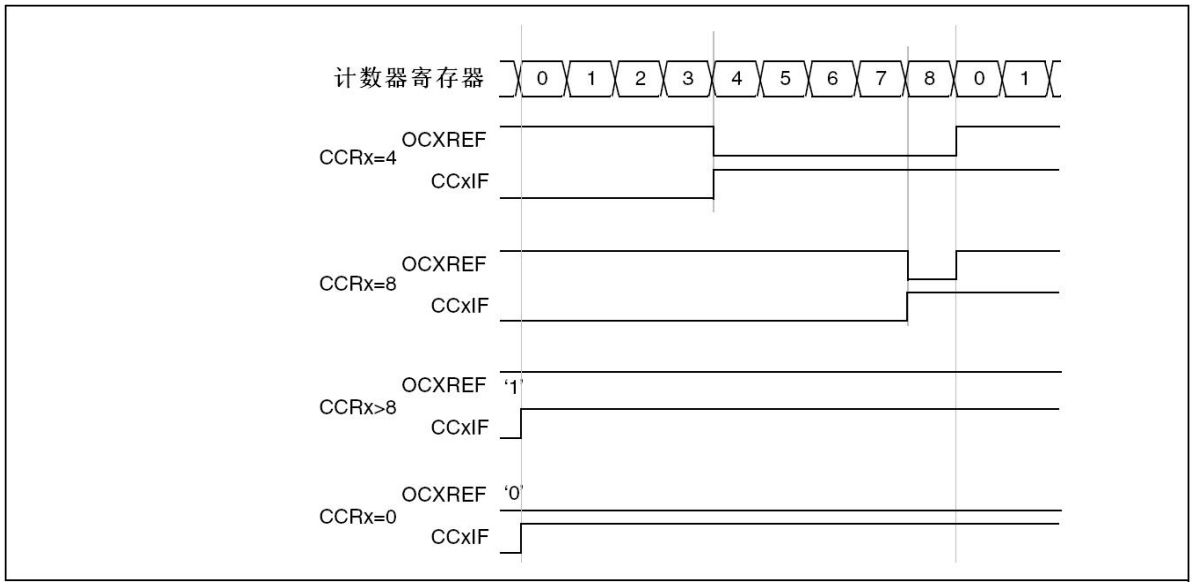
PWM 边沿对齐模式

向上计数配置

当 TIMx_CR1 寄存器中的 DIR 位为低的时候执行向上计数。参看 14.3.2 节。

下面是一个 PWM 模式 1 的例子。当 TIMx_CNT<TIMx_CCRx 时 PWM 信号参考 OCxREF 为高，否 则为低。如果 TIMx_CCRx 中的比较值大于自动重装载值(TIMx_ARR)，则 OCxREF 保持为’1’。如果比较值为 0，则 OCxREF 保持为’0’。下图为 TIMx_ARR=8 时边沿对齐的 PWM 波形实例。

图 115 边沿对齐的 PWM 波形(ARR=8)



向下计数的配置

当 TIMx_CR1 寄存器的 DIR 位为高时执行向下计数。参看 14.3.2 节。

在PWM 模式 1 ， 当 TIMx_CNT>TIMx_CCRx 时 参 考 信 号OCxREF 为 低 ， 否 则 为 高 。 如果 TIMx_CCRx 中的比较值大于 TIMx_ARR 中的自动重装载值，则 OCxREF 保持为’1’。该模式下不 能产生 0%的 PWM 波形。

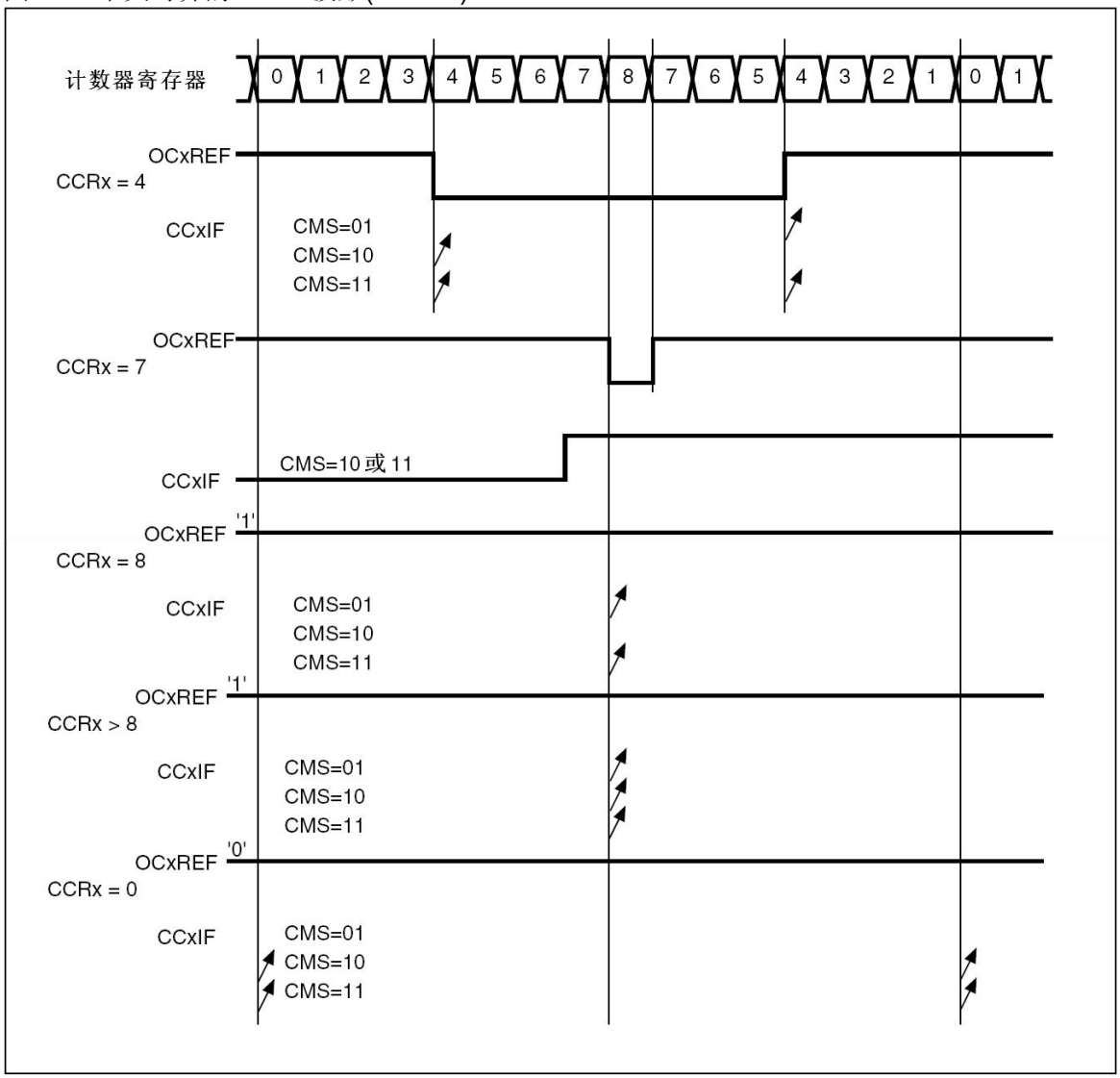
PWM 中央对齐模式

当TIMx_CR1 寄存器中的CMS 位不为’00’时，为中央对齐模式(所有其他的配置对 OCxREF/OCx 信号都有相同的作用)。根据不同的 CMS 位设置，比较标志可以在计数器向上计数时被置’1’、在计数器向下计数时被置’1’、或在计数器向上和向下计数时被置’1’。TIMx_CR1 寄存器中的计数方向位(DIR)由硬件更新，不要用软件修改它。参看 14.3.2 节的中央对齐模式。

下图给出了一些中央对齐的 PWM 波形的例子

- TIMx_ARR=8
- PWM 模式 1
- TIMx_CR1 寄存器中的 CMS=01，在中央对齐模式 1 时，当计数器向下计数时设置比较标志。

图 116 中央对齐的 PWM 波形(APR=8)



使用中央对齐模式的提示：

- 进入中央对齐模式时，使用当前的向上/向下计数配置：这就意味着计数器向上还是向下计数取决于 TIMx_CR1 寄存器中 DIR 位的当前值。此外，软件不能同时修改 DIR 和 CMS 位。
- 不推荐当运行在中央对齐模式时改写计数器，因为这会产生不可预知的结果。特别地：
  - 如果写入计数器的值大于自动重加载的值(TIMx_CNT>TIMx_ARR)，则方向不会被更新。例如，如果计数器正在向上计数，它就会继续向上计数。
  - 如果将 0 或者 TIMx_ARR 的值写入计数器，方向被更新，但不产生更新事件 UEV。
- 使用中央对齐模式最保险的方法，就是在启动计数器之前产生一个软件更新(设置 TIMx_EGR 位中的 UG 位)，不要在计数进行过程中修改计数器的值。

13.3.10 单脉冲模式

单脉冲模式(OPM)是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后，产生一个脉宽可程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TIMx_CR1 寄存器中的 OPM 位将选择单脉冲模式，这样可以让计数器自动地在产生下一个更新事件 UEV 时停止。

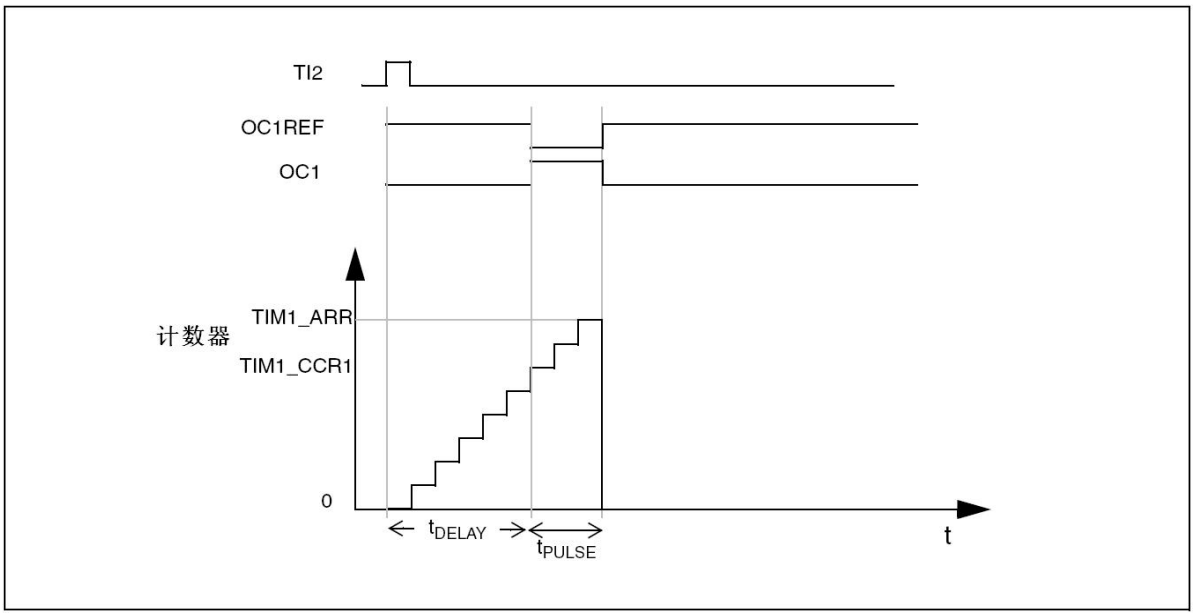
仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前(当定时器正在等待触发)，必

须如下配置：

向上计数方式：CNT < CCRx ≤ ARR (特别地，0 < CCRx)，

向下计数方式：CNT > CCRx。

图 117 单脉冲模式的例子



例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 t_{DELAY} 之后，在 OC1 上产生一个长度为 t_{PULSE} 的正脉冲。

假定 TI2FP2 作为触发 1:

- 置 TIMx_CCMR1 寄存器中的 CC2S='01'，把 TI2FP2 映像到 TI2。
- 置 TIMx_CCER 寄存器中的 CC2P='0'，使 TI2FP2 能够检测上升沿。
- 置 TIMx_SMCR 寄存器中的 TS='110'，TI2FP2 作为从模式控制器的触发(TRGI)。
- 置 TIMx_SMCR 寄存器中的 SMS='110'(触发模式)，TI2FP2 被用来启动计数器。

OPM 波形由写入比较寄存器的数值决定(要考虑时钟频率和计数器预分频器)

- t_{DELAY} 由写入 TIMx_CCR1 寄存器中的值定义。
- t_{PULSE} 由自动装载值和比较值之间的差值定义(TIMx_ARR - TIMx_CCR1)。
- 假定当发生比较匹配时要产生从'0'到'1'的波形，当计数器到达预装载值时要产生一个从'1'到'0'的波形；首先要置 TIMx_CCMR1 寄存器的 OC1M='111'，进入 PWM 模式 2；根据需要有选择地使能预装载寄存器：置 TIMx_CCMR1 中的 OC1PE='1'和 TIMx_CR1 寄存器中的 ARPE；然后在 TIMx_CCR1 寄存器中填写比较值，在 TIMx_ARR 寄存器中填写自动装载 值，修改 UG 位来产生一个更新事件，然后等待在 TI2 上的一个外部触发事件。本例中， CC1P='0'。

在这个例子中，TIMx_CR1 寄存器中的 DIR 和 CMS 位应该置低。

因为只需一个脉冲，所以必须设置 TIMx_CR1 寄存器中的 OPM='1'，在下一个更新事件(当计数器从自动装载值翻转到 0)时停止计数。

特殊情况：OCx 快速使能：

在单脉冲模式下，在 TIx 输入脚的边沿检测逻辑设置 CEN 位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期，因此它限制了可得到的最小延时 t_{DELAY}。 如果要以最小延时输出波形，可以设置 TIMx_CCMRx 寄存器中的 OCxFE 位；此时 OCxREF(和OCx)被强制响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。OCxFE 只在通道配置为 PWM1 和 PWM2 模式时起作用。

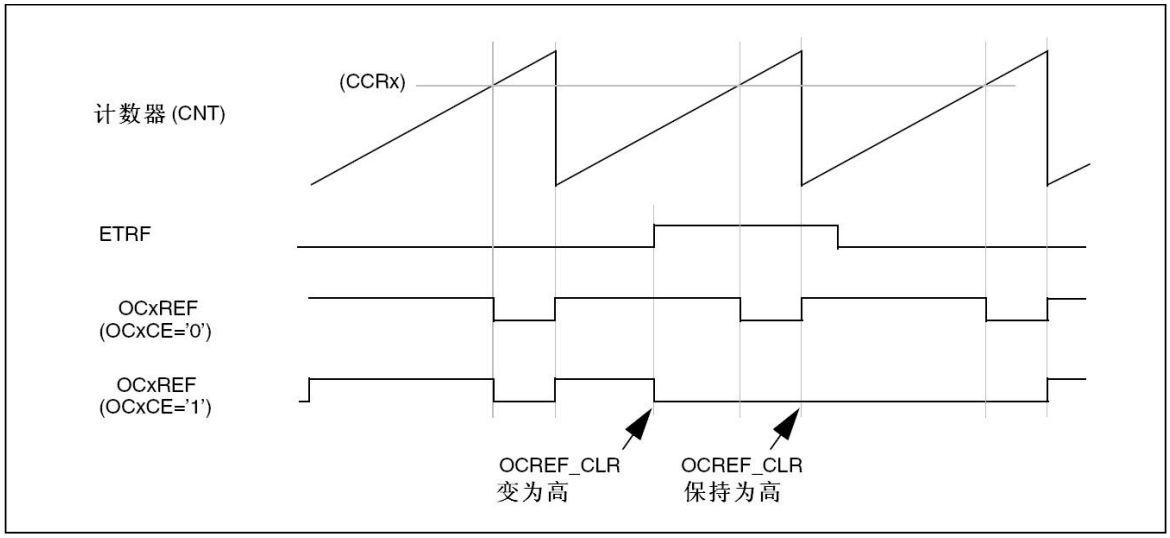
13.3.11 在外部事件时清除 OCxREF 信号

对于一个给定的通道，设置 TIMx_CCMRx 寄存器中对应的 OCxCE 位为'1'，能够用 ETRF 输入端 的高电平把 OCxREF 信号拉低，OCxREF 信号将保持为低直到发生下一次的更新事件 UEV。该功能只能用于输出比较和 PWM 模式，而不能用于强置模式。

例如，OCxREF 信号可以联到一个比较器的输出，用于控制电流。这时，ETR 必须配置如下：

1. 外部触发预分频器必须处于关闭：TIMx_SMCR 寄存器中的 ETPS[1:0]='00'。
2. 必须禁止外部时钟模式 2：TIMx_SMCR 寄存器中的 ECE='0'。
3. 外部触发极性(ETP)和外部触发滤波器(ETF)可以根据需要配置。 下图显示了当 ETRF 输入变为高时，对应不同 OCxCE 的值，OCxREF 信号的动作。在这个例子中，定时器 TIMx 被置于 PWM 模式。

图 118 清除 TIMx 的 OCxREF



13.3.12 编码器接口模式

选择编码器接口模式的方法是：如果计数器只在 TI2 的边沿计数，则置 TIMx_SMCR 寄存器中的SMS=001；如果只在 TI1 边沿计数，则置 SMS=010；如果计数器同时在 TI1 和 TI2 边沿计数，则置 SMS=011。

通过设置 TIMx_CCER 寄存器中的 CC1P 和 CC2P 位，可以选择 TI1 和 TI2 极性；如果需要，还可以 对输入滤波器编程。

两个输入 TI1 和 TI2 被用来作为增量编码器的接口。参看表 55，假定计数器已经启动(TIMx_CR1 寄存器中的 CEN='1')，计数器由每次在 TI1FP1 或 TI2FP2 上的有效跳变驱动。TI1FP1 和 TI2FP2 是 TI1 和 TI2 在通过输入滤波器和极性控制后的信号；如果没有滤波和变相，则 TI1FP1=TI1，TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号 的跳变顺序，计数器向上或向下计数，同时硬件对 TIMx_CR1 寄存器的 DIR 位进行相应的设置。不管计数器是依靠 TI1 计数、依靠 TI2 计数或者同时依靠 TI1 和 TI2 计数。在任一输入端(TI1 或者TI2)的跳变都会重新计算 DIR 位。

编码器接口模式基本上相当于使用了一个带有方向选择的外部时钟。这意味着计数器只在 0 到TIMx_ARR 寄存器的自动装载值之间连续计数(根据方向，或是 0 到 ARR 计数，或是 ARR 到 0 计数)。所以在开始计数之前必须配置 TIMx_ARR；同样，捕获器、比较器、预分频器、触发输出特性等仍工作如常。

在这个模式下，计数器依照增量编码器的速度和方向被自动的修改，因此计数器的内容始终指示着编码器的位置。计数方向与相连的传感器旋转的方向对应。下表列出了所有可能的组合，假设 TI1 和 TI2 不同时变换。

表 55 计数方向与编码器信号的关系

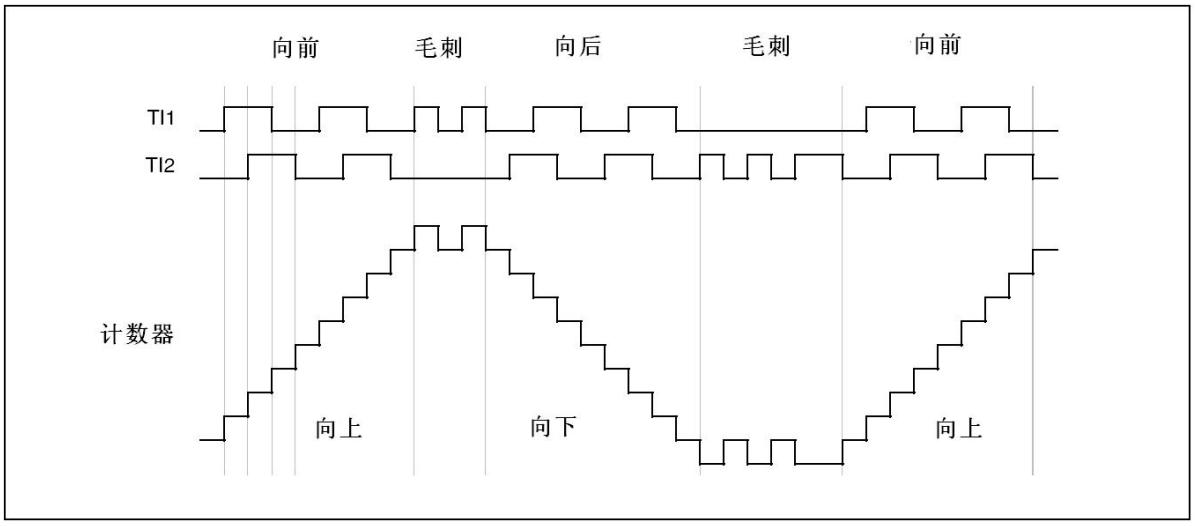
有效边沿	相对信号的电平 (TI1FP1 对应 TI2, TI2FP2 对	TI1FP1 信号		TI2FP2 信号	
		上升	下降	上升	下降
仅在 TI1 计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在 TI2 计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在 TI1 和 TI2 上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

一个外部的增量编码器可以直接与 MCU 连接而不需要外部接口逻辑。但是，一般会使用比较器将编码器的差动输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信号表示机械零点，可以把它连接到一个外部中断输入并触发一个计数器复位。

下图是一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；抖动可能会在传感器的位置靠近一个转换点时产生。在这个例子中，我们假定配置如下：

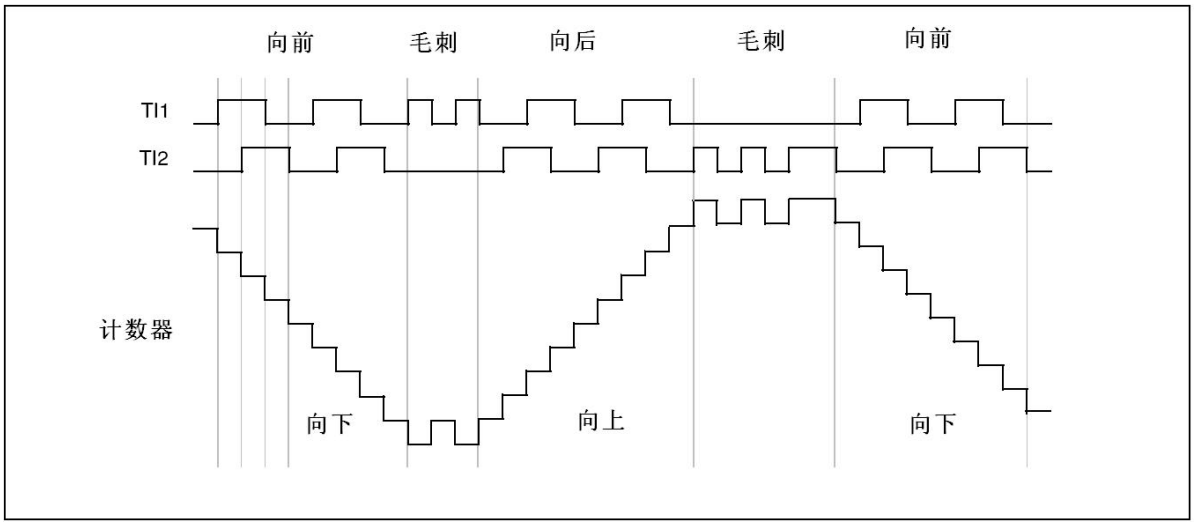
- CC1S=’01’ (TIMx_CCMR1 寄存器，IC1FP1 映射到 TI1)
- CC2S=’01’ (TIMx_CCMR2 寄存器，IC2FP2 映射到 TI2)
- CC1P=’0’(TIMx_CCER 寄存器，IC1FP1 不反相，IC1FP1=TI1)
- CC2P=’0’(TIMx_CCER 寄存器，IC2FP2 不反相，IC2FP2=TI2)
- SMS=’011’(TIMx_SMCR 寄存器，所有的输入均在上升沿和下降沿有效).
- CEN=’1’(TIMx_CR1 寄存器，计数器使能)

图 119 编码器模式下的计数器操作实例



下图为当 IC1FP1 极性反相时计数器的操作实例(CC1P='1',其他配置与上例相同)

图 120 IC1FP1 反相的编码器接口模式实例



当定时器配置成编码器接口模式时，提供传感器当前位置的信息。使用第二个配置在捕获模式的定时器，可以测量两个编码器事件的间隔，获得动态的信息(速度，加速度，减速度)。指示机械零点的编码器输出可被用做此目的。根据两个事件间的间隔，可以按照固定的时间读出计数器。如果可能的话，你可以把计数器的值锁存到第三个输入捕获寄存器(捕获信号必须是周期的并且可以由另一个定时器产生)；也可以通过一个由实时时钟产生的 DMA 请求来读取它的值。

13.3.13 定时器输入异或功能

TIMx_CR2 寄存器中的 TI1S 位，允许通道 1 的输入滤波器连接到一个异或门的输出端，异或门的3 个输入端为 TIMx_CH1、TIMx_CH2 和 TIMx_CH3。
异或输出能够被用于所有定时器的输入功能，如触发或输入捕获。上一章 13.3.18 节给出了此特性用于连接霍尔传感器的例子。

13.3.14 定时器和外部触发的同步

TIMx 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

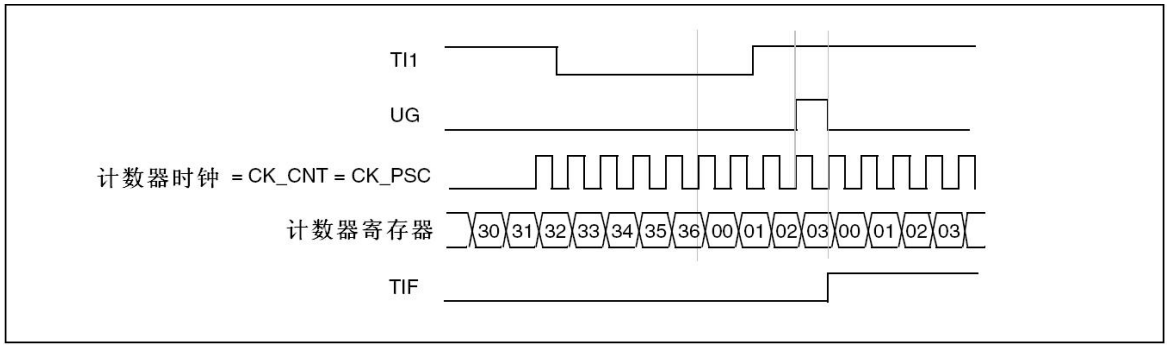
从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIMx_CR1 寄存器的 URS 位为低，还会产生一个更新事件 UEV；然后所有的预装载寄存器(TIMx_ARR，TIMx_CCRx)都会被更新。
在下面的例子中，TI1 输入端的上升沿导致向上计数器被清零：

- 配置通道 1 以检测 TI1 的上升沿。配置输入滤波器的带宽(在本例中，不需要任何滤波器，因 此保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置它。CC1S 位只选 择输入捕获源，即 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=0 以 确定极性(只检测上升沿)。
- 置 TIMx_SMCR 寄存器中 SMS=100，配置定时器为复位模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1 出现一个上升沿；此时，计数器被清零然后从 0 重新开始计数。同时，触发标志(TIMx_SR 寄存器中的 TIF 位)被设置，根据 TIMx_DIER 寄存器中 TIE(中断使能)位和 TDE(DMA 使能)位的设置，产生一个中断请求或一个 DMA 请求。
下图显示当自动重装载寄存器 TIMx_ARR=0x36 时的动作。在 TI1 上升沿和计数器的实际复位之间的延时，取决于 TI1 输入端的重同步电路。

图 121 复位模式下的控制电路



从模式：门控模式

按照选中的输入端电平使能计数器。

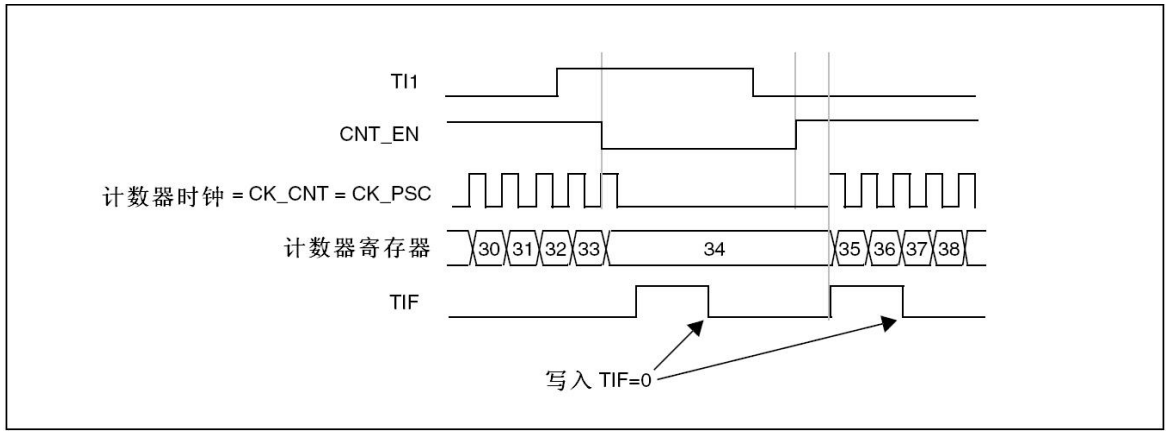
在如下的例子中，计数器只在 TI1 为低时向上计数：

- 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽(本例中，不需要滤波，所以保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入 捕获源，置 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=1 以确定极 性(只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=0，则计数器不能 启动，不论触发输入电平如何。

只要 TI1 为低，计数器开始依据内部时钟计数，在 TI1 变高时停止计数。当计数器开始或停止时都设置 TIMx_SR 中的 TIF 标置。

TI1 上升沿和计数器实际停止之间的延时，取决于 TI1 输入端的重同步电路。

图 122 门控模式下的控制电路



从模式：触发模式

输入端上选中的事件使能计数器。

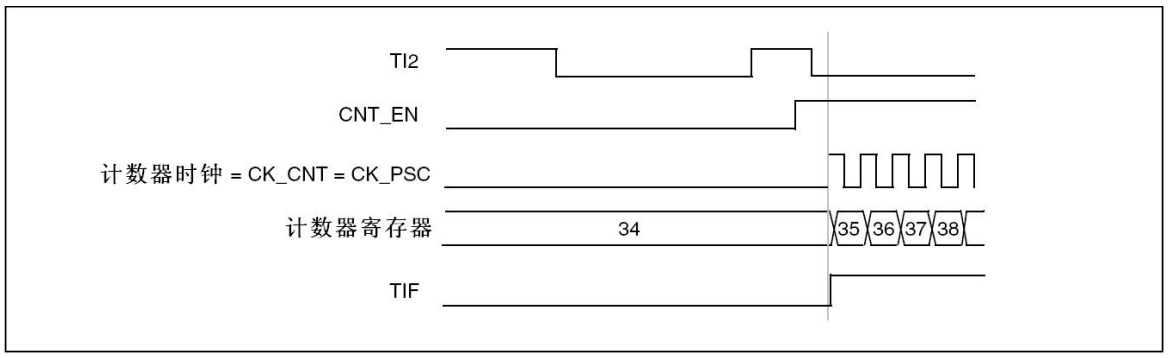
在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

- 配置通道 2 检测 TI2 的上升沿。配置输入滤波器带宽(本例中，不需要任何滤波器，保持 IC2F=0000)。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕 获源，置 TIMx_CCMR1 寄存器中 CC2S=01。置 TIMx_CCER 寄存器中 CC2P=1 以确定极性 (只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIMx_SMCR 寄存器中 TS=110，选择 TI2 作为输入源。 当 TI2 出现一个上升沿时，计数器开始在内部时钟驱动下

计数，同时设置 TIF 标志。

TI2 上升沿和计数器启动计数之间的延时，取决于 TI2 输入端的重同步电路。

图 123 触发器模式下的控制电路



从模式：外部时钟模式 2 ＋ 触发模式

外部时钟模式 2 可以与另一种从模式(外部时钟模式 1 和编码器模式除外)一起使用。这时，ETR 信号被用作外部时钟的输入，在复位模式、门控模式或触发模式时可以选择另一个输入作为触发输入。不建议使用 TIMx_SMCR 寄存器的 TS 位选择 ETR 作为 TRGI。

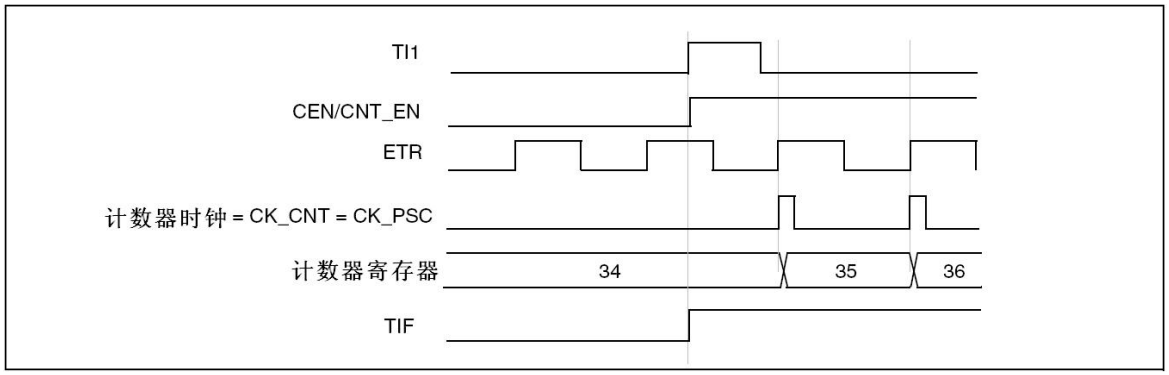
下面的例子中，TI1 上出现一个上升沿之后，计数器即在 ETR 的每一个上升沿向上计数一次：

1. 通过 TIMx_SMCR 寄存器配置外部触发输入电路：
   - ETF=0000：没有滤波
   - ETPS=00：不用预分频器
   - ETP=0：检测 ETR 的上升沿，置 ECE=1 使能外部时钟模式
2. 按如下配置通道 1，检测 TI 的上升沿：
   - IC1F=0000：没有滤波
   - 触发操作中不使用捕获预分频器，不需要配置
   - 置 TIMx_CCMR1 寄存器中 CC1S=01，选择输入捕获源
   - 置 TIMx_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)
3. 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式。置 TIMx_SMCR 寄存器中
   TS=101，选择 TI1 作为输入源。

当 TI1 上出现一个上升沿时，TIF 标志被设置，计数器开始在 ETR 的上升沿计数。

ETR 信号的上升沿和计数器实际复位间的延时，取决于 ETRP 输入端的重同步电路。

图 124 外部时钟模式 2＋触发模式下的控制电路



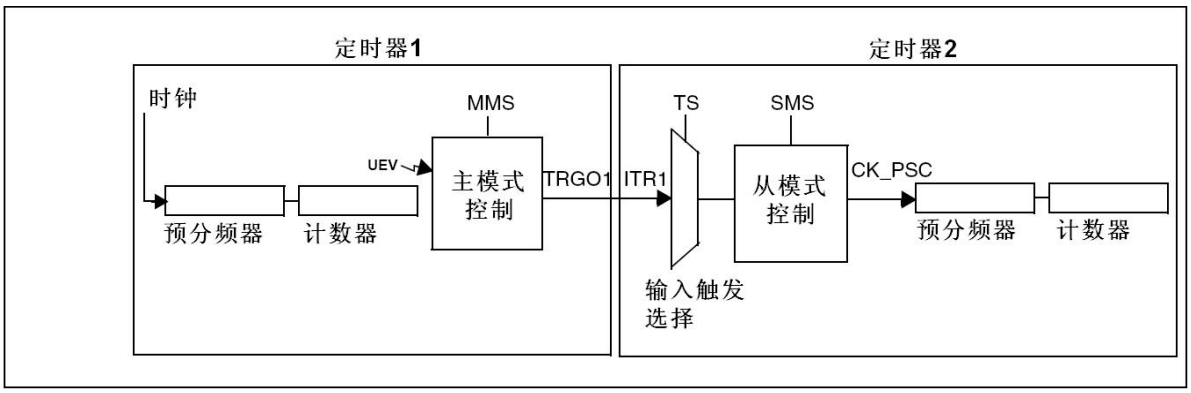
13.3.15 定时器同步

所有 TIMx 定时器在内部相连，用于定时器同步或链接。当一个定时器处于主模式时，它可以对另一个处于从模式的定时器的计数器进行复位、启动、停止或提供时钟等操作。

下图显示了触发选择和主模式选择模块的概况。

使用一个定时器作为另一个定时器的预分频器

图 125 主/从定时器的例子



如：可以配置定时器 1 作为定时器 2 的预分频器。参考图 125，进行下述操作：

- 配置定时器 1 为主模式，它可以在每一个更新事件 UEV 时输出一个周期性的触发信号。在 TIM1_CR2 寄存器的 MMS=’010’时，每当产生一个更新事件时在 TRGO1 上输出一个上升沿信号。
- 连接定时器 1 的 TRGO1 输出至定时器 2，设置 TIM2_SMCR 寄存器的 TS=’000’，配置定时器 2 为使用 ITR1 作为内部触发的从模式。
- 然后把从模式控制器置于外部时钟模式 1(TIM2_SMCR 寄存器的 SMS=111)；这样定时器 2 即可由定时器 1 周期性的上升沿(即定时器 1 的计数器溢出)信号驱动。
- 最后，必须设置相应(TIMx_CR1 寄存器)的 CEN 位分别启动两个定时器。

注：　如果 OCx 已被选中为定时器 1 的触发输出(MMS=1xx)，它的上升沿用于驱动定时器 2 的计数器。

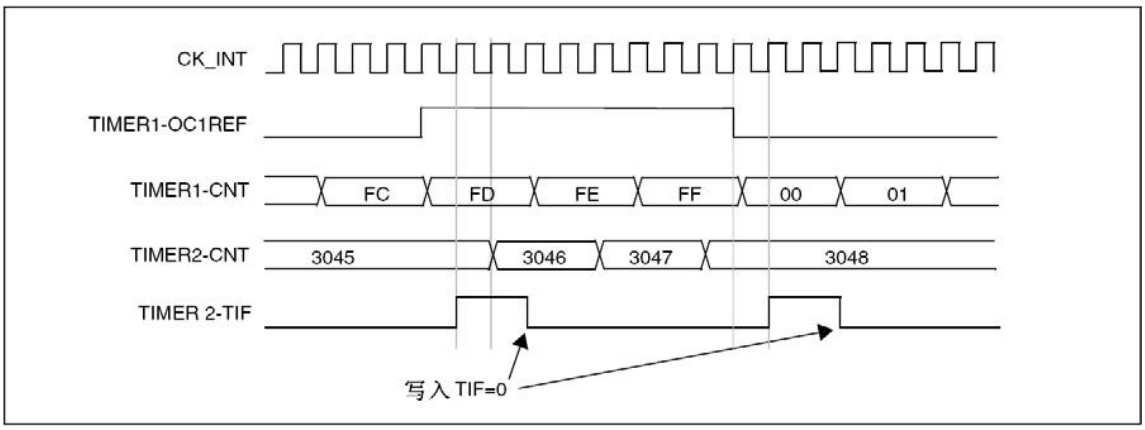
使用一个定时器使能另一个定时器

在这个例子中，定时器 2 的使能由定时器 1 的输出比较控制。参考图 125 的连接。只当定时器 1 的 OC1REF 为高时，定时器 2 才对分频后的内部时钟计数。两个定时器的时钟频率都是由预分频器对 CK_INT 除以 3($f_{CK\_CNT}=f_{CK\_INT}/3$)得到。

- 配置定时器 1 为主模式，送出它的输出比较参考信号(OC1REF)为触发输出(TIM1_CR2 寄存器的 MMS=100)
- 配置定时器 1 的 OC1REF 波形(TIM1_CCMR1 寄存器)
- 配置定时器 2 从定时器 1 获得输入触发(TIM2_SMCR 寄存器的 TS=000)
- 配置定时器 2 为门控模式(TIM2_SMCR 寄存器的 SMS=101)
- 置 TIM2_CR1 寄存器的 CEN=1 以使能定时器 2
- 置 TIM1_CR1 寄存器的 CEN=1 以启动定时器 1

注：　定时器 2 的时钟不与定时器 1 的时钟同步，这个模式只影响定时器 2 计数器的使能信号。

图 126 定时器 1 的 OC1REF 控制定时器 2

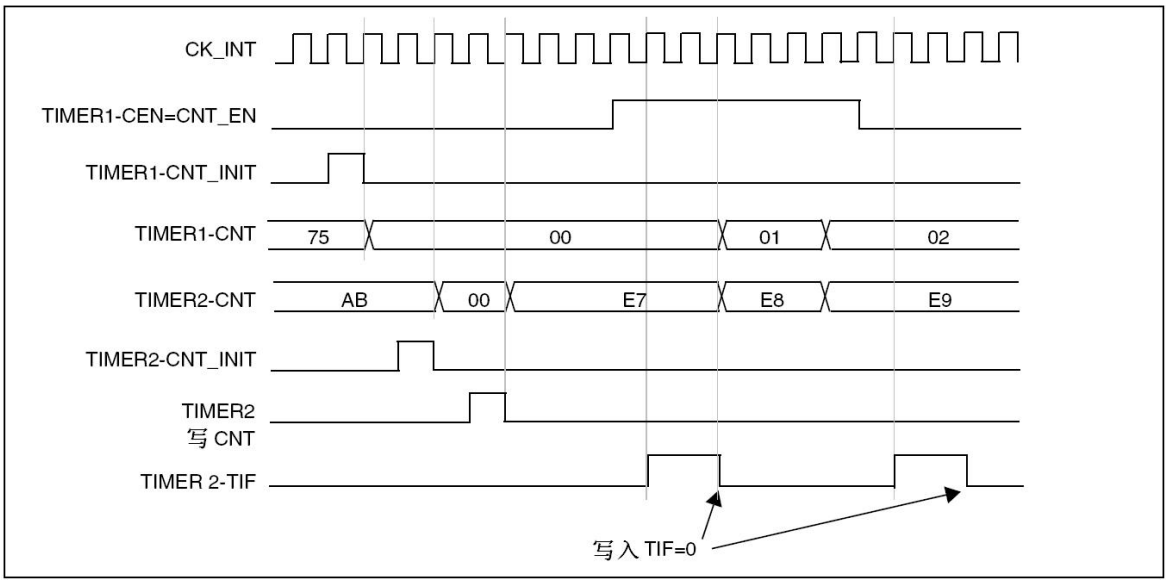


在图 126 的例子中，在定时器 2 启动之前，它们的计数器和预分频器未被初始化，因此它们从当前的数值开始计数。可以在启动定时器 1 之前复位 2 个定时器，使它们从给定的数值开始，即在定时器计数器中写入需要的任意数值。写 TIMx_EGR 寄存器的 UG 位即可复位定时器。

在下一个例子中，需要同步定时器 1 和定时器 2。定时器 1 是主模式并从 0 开始，定时器 2 是从模式并从 0xE7 开始；2 个定时器的预分频器系数相同。写'0'到 TIM1_CR1 的 CEN 位将禁止定时器 1， 定时器 2 随即停止。

- 配置定时器 1 为主模式，送出输出比较 1 参考信号(OC1REF)做为触发输出(TIM1_CR2 寄存 器的 MMS=100)。
- 配置定时器 1 的 OC1REF 波形(TIM1_CCMR1 寄存器)。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2_SMCR 寄存器的 TS=000)
- 配置定时器 2 为门控模式(TIM2_SMCR 寄存器的 SMS=101)
- 置 TIM1_EGR 寄存器的 UG='1'，复位定时器 1。
- 置 TIM2_EGR 寄存器的 UG='1'，复位定时器 2。
- 写'0xE7'至定时器 2 的计数器(TIM2_CNTL)，初始化它为 0xE7。
- 置 TIM2_CR1 寄存器的 CEN='1'以使能定时器 2。
- 置 TIM1_CR1 寄存器的 CEN='1'以启动定时器 1。
- 置 TIM1_CR1 寄存器的 CEN=' 0' 以停止定时器 1。

图 127 通过使能定时器 1 可以控制定时器 2

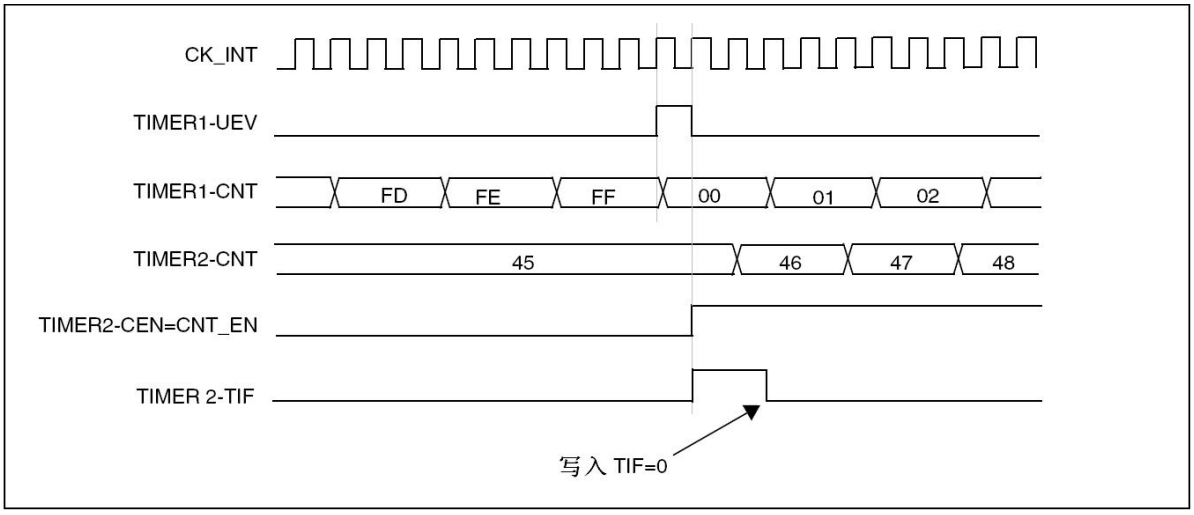


使用一个定时器去启动另一个定时器

在这个例子中，使用定时器 1 的更新事件使能定时器 2。参考图 125 的连接。一旦定时器 1 产生更 新事件，定时器 2 即从它当前的数值(可以是非 0)按照分频的内部时钟开始计数。在收到触发信号时，定时器 2 的 CEN 位被自动地置'1'，同时计数器开始计数直到写'0'到 TIM2_CR1 寄存器的 CEN 位。两个定时器的时钟频率都是由预分频器对 CK_INT 除以 3($f_{CK\_CNT}=f_{CK\_INT}/3$)。

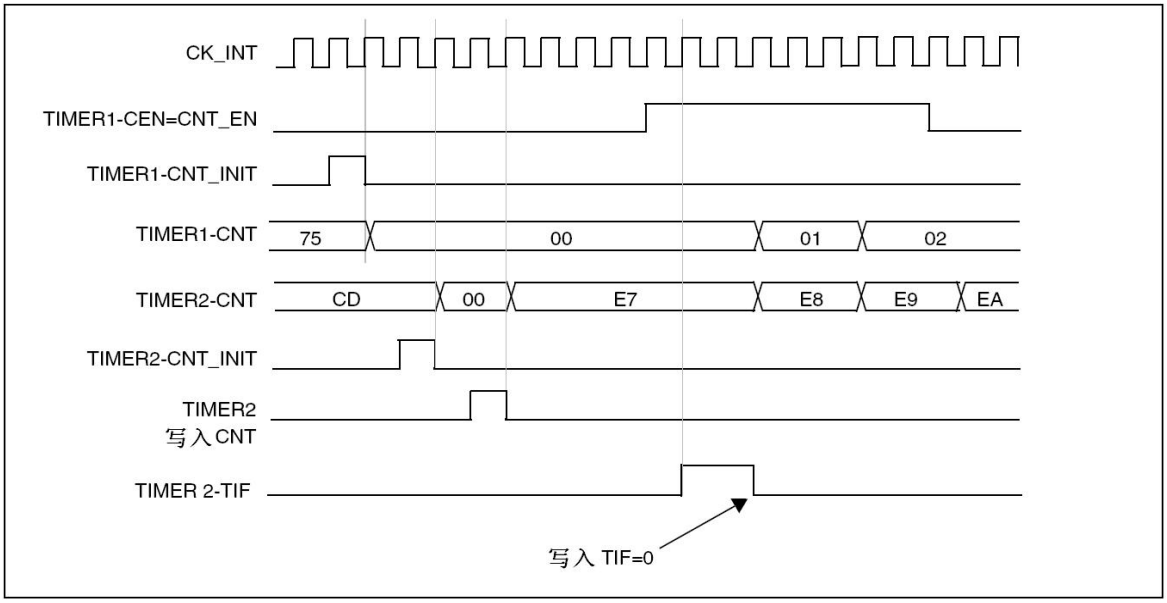
- 配置定时器 1 为主模式，送出它的更新事件(UEV)做为触发输出(TIM1_CR2 寄存器的 MMS=010)。
- 配置定时器 1 的周期(TIM1_ARR 寄存器)。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2_SMCR 寄存器的 TS=000)
- 配置定时器 2 为触发模式(TIM2_SMCR 寄存器的 SMS=110)
- 置 TIM1_CR1 寄存器的 CEN=1 以启动定时器 1。

图 128 使用定时器 1 的更新触发定时器 2



在上一个例子中，可以在启动计数之前初始化两个计数器。图 129 显示在与 0 相同配置情况下，使用触发模式而不是门控模式(TIM2_SMCR 寄存器的 SMS=110)的动作。

图 129 利用定时器 1 的使能触发定时器 2



使用一个定时器作为另一个的预分频器

这个例子使用定时器 1 作为定时器 2 的预分频器。参考图 125 的连接，配置如下：

- 配置定时器 1 为主模式，送出它的更新事件 UEV 做为触发输出(TIM1_CR2 寄存器的 MMS='010')。然后每次计数器溢出时输出一个周期信号。
- 配置定时器 1 的周期(TIM1_ARR 寄存器)。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2_SMCR 寄存器的 TS=000)
- 配置定时器 2 使用外部时钟模式(TIM2_SMCR 寄存器的 SMS=111)
- 置 TIM1_CR2 寄存器的 CEN=1 以启动定时器 2。
- 置 TIM1_CR1 寄存器的 CEN=1 以启动定时器 1。

使用一个外部触发同步地启动 2 个定时器

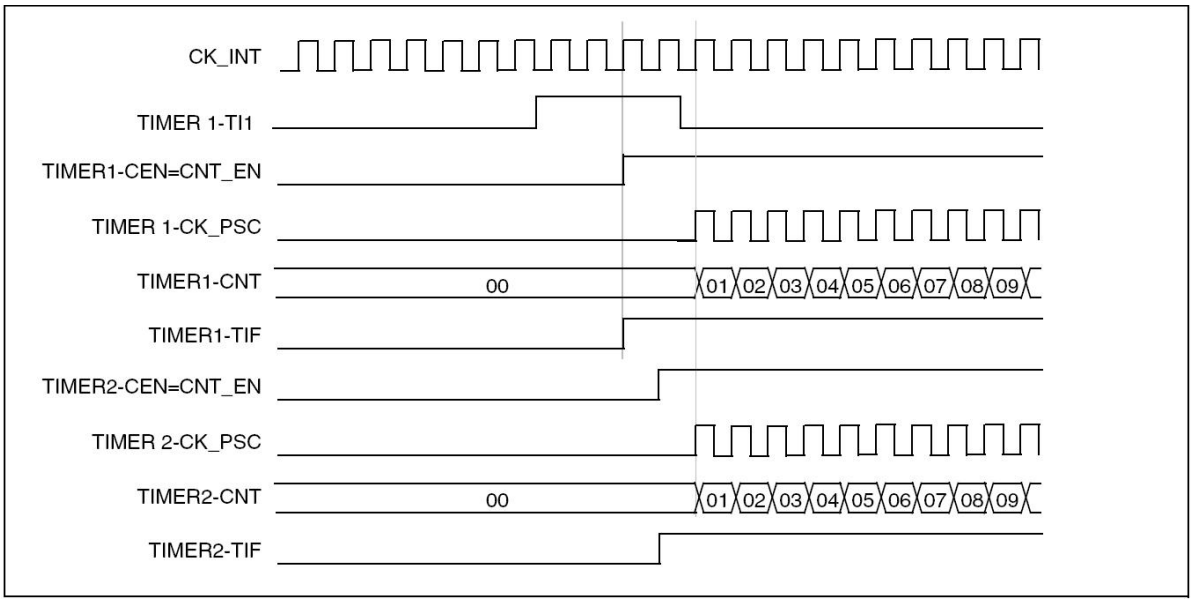
这个例子中当定时器 1 的 TI1 输入上升时使能定时器 1，使能定时器 1 的同时使能定时器 2，参见图 125。为保证计数器的对齐，定时器 1 必须配置为主/从模式(对应 TI1 为从，对应定时器 2 为主)：

- 配置定时器 1 为主模式，送出它的使能做为触发输出(TIM1_CR2 寄存器的 MMS='001')
- 配置定时器 1 为从模式，从 TI1 获得输入触发(TIM1_SMCR 寄存器的 TS='100')。
- 配置定时器 1 为触发模式(TIM1_SMCR 寄存器的 SMS='110')。
- 配置定时器 1 为主/从模式，TIM1_SMCR 寄存器的 MSM='1'。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2_SMCR 寄存器的 TS=000)
- 配置定时器 2 为触发模式(TIM2_SMCR 寄存器的 SMS='110')。

当定时器 1 的 TI1 上出现一个上升沿时，两个定时器同步地按照内部时钟开始计数，两个 TIF 标志也同时被设置。

注：在这个例子中，在启动之前两个定时器都被初始化(设置相应的 UG 位)，两个计数器都从 0 开始，但可以通过写入任意一个计数器寄存器(TIMx_CNT)在定时器间插入一个偏移。下图中能看到主/从模式下在定时器 1 的 CNT_EN 和 CK_PSC 之间有个延迟。

图 130 使用定时器 1 的 TI1 输入触发定时器 1 和定时器 2



13.3.16 调试模式

当微控制器进入调试模式(Cortex-M3 核心停止)，根据 DBG 模块中 DBG_TIMx_STOP 的设置，TIMx 计数器或者继续正常操作，或者停止。详见随后第 25.15.2 节：支持定时器、看门狗、bxCAN 和 I^2C 的调试。

13.4 TIMx 寄存器描述

关于在寄存器描述里面所用到的缩写，详见第 2.1 节。 可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

13.4.1 控制寄存器 1(TIMx_CR1)

偏移地址：0x00 复位值：

0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:10	保留，始终读为 0。
位 9:8	**CKD[1:0]**: 时钟分频因子(Clock division) 定义在定时器时钟(CK_INT)频率与数字滤波器(ETR，TIx)使用的采样频率之间的分频比例。 00：$t_{DTS} = t_{CK\_INT}$ 01：$t_{DTS} = 2 \times t_{CK\_INT}$ 10：$t_{DTS} = 4 \times t_{CK\_INT}$ 11：保留
位 7	**ARPE**：自动重装载预装载允许位(Auto-reload preload enable) 0： TIMx_ARR 寄存器没有缓冲； 1：TIMx_ARR 寄存器被装入缓冲器。
位 6:5	**CMS[1:0]**：选择中央对齐模式(Center-aligned mode selection) 00：边沿对齐模式。计数器依据方向位(DIR)向上或向下计数。 01：中央对齐模式 1。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位，只在计数器向下计数时被设置。 10：中央对齐模式 2。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx 寄存器 中 CCxS=00)的输出比较中断标志位，只在计数器向上计数时被设置。 11：中央对齐模式 3。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位，在计数器向上和向下计数时均被设置。 注：在计数器开启时(CEN=1)，不允许从边沿对齐模式转换到中央对齐模式。
位 4	**DIR**：方向(Direction) 0：计数器向上计数； 1：计数器向下计数。 注：当计数器配置为中央对齐模式或编码器模式时，该位为只读。
位 3	**OPM**：单脉冲模式(One pulse mode) 0：在发生更新事件时，计数器不停止； 1：在发生下一次更新事件(清除 CEN 位)时，计数器停止。

位 2	**URS**：更新请求源(Update request source) 软件通过该位选择 UEV 事件的源 0：如果使能了更新中断或 DMA 请求，则下述任一事件产生更新中断或 DMA 请求： – 计数器溢出/下溢 – 设置 UG 位 – 从模式控制器产生的更新 1：如果使能了更新中断或 DMA 请求，则只有计数器溢出/下溢才产生更新中断或 DMA 请求。
位 1	**UDIS**：禁止更新(Update disable) 软件通过该位允许/禁止 UEV 事件的产生 0：允许 UEV。更新(UEV)事件由下述任一事件产生： – 计数器溢出/下溢 – 设置 UG 位 – 从模式控制器产生的更新 具有缓存的寄存器被装入它们的预装载值。(译注：更新影子寄存器) 1：禁止 UEV。不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
位 0	**CEN**：使能计数器 0：禁止计数器； 1：使能计数器。 注：在软件设置了 CEN 位后，外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。 在单脉冲模式下，当发生更新事件时，CEN 被自动清除。

13.4.2 控制寄存器 2(TIMx_CR2)

偏移地址：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								TI1S	MMS[2:0]			CCDS	保留		
								rw	rw	rw	rw	rw			

位 15:8	保留，始终读为 0。
位 7	**TI1S**：TI1 选择(TI1 selection) 0：TIMx_CH1 引脚连到 TI1 输入； 1：TIMx_CH1、TIMx_CH2 和 TIMx_CH3 引脚经异或后连到 TI1 输入。 见上一章 12.3.18 的与霍尔传感器的接口一节。
位 6:4	**MMS[2:0]**：主模式选择(Master mode selection) 这 3 位用于选择在主模式下送到从定时器的同步信息(TRGO)。可能的组合如下： 000：**复位** – TIMx_EGR 寄存器的 UG 位被用于作为触发输出(TRGO)。如果是触发输入产生复位(从模式控制器处于复位模式)，则 TRGO 上的信号相对实际的复位会有一个延迟。 001：**使能** – 计数器使能信号 CNT_EN 被用于作为触发输出(TRGO)。有时需要在同一时间启动多个定时器或控制在一段时间内使能从定时器。计数器使能信号是通过 CEN 控制位和门控模式下的触发输入信号的逻辑或产生。 当计数器使能信号受控于触发输入时，TRGO 上会有一个延迟，除非选择了主/从模式 TIMx_SMCR 寄存器中 MSM 位的描述)。 010：**更新** – 更新事件被选为触发输入(TRGO)。例如，一个主定时器的时钟可以被用作一个从定时器的预分频器。 011：**比较脉冲** – 在发生一次捕获或一次比较成功时，当要设置 CC1IF 标志时(即使它已经高)，触发输出送出一个正脉冲(TRGO)。 100：**比较**– OC1REF 信号被用于作为触发输出(TRGO)。

	101：**比较**– OC2REF 信号被用于作为触发输出(TRGO)。 110：**比较**– OC3REF 信号被用于作为触发输出(TRGO)。 111：**比较**– OC4REF 信号被用于作为触发输出(TRGO)。
位 3	**CCDS**：捕获/比较的 DMA 选择 0：当发生 CCx 事件时，送出 CCx 的 DMA 请求； 1：当发生更新事件时，送出 CCx 的 DMA 请求。
位 2:0	保留，始终读为 0。

13.4.3 从模式控制寄存器(TIMx_SMCR)

偏移地址：0x08

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			保留	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

位 15	**ETP**：外部触发极性(External trigger polarity) 该位选择是用 ETR 还是 ETR 的反相来作为触发操作 0：ETR 不反相，高电平或上升沿有效； 1：ETR 被反相，低电平或下降沿有效。
位 14	**ECE**：外部时钟使能位(External clock enable) 该位启用外部时钟模式 2 0：禁止外部时钟模式 2； 1：使能外部时钟模式 2。计数器由 ETRF 信号上的任意有效边沿驱动。 注 1：设置 ECE 位与选择外部时钟模式 1 并将 TRGI 连到 ETRF(SMS=111 和 TS=111)具有相同功效。 注 2：下述从模式可以与外部时钟模式 2 同时使用：复位模式、门控模式和触发模式；但是，这时 TRGI 不能连到 ETRF(TS 位不能是'111')。 注 3：外部时钟模式 1 和外部时钟模式 2 同时被使能时，外部时钟的输入是 ETRF。
位 13:12	**ETPS[1:0]**：外部触发预分频(External trigger prescaler) 外部触发信号 ETRP 的频率必须最多是 CK_INT 频率的 1/4。当输入较快的外部时钟时，可以使用预分频降低 ETRP 的频率。 00：关闭预分频； 01：ETRP 频率除以 2； 10：ETRP 频率除以 4； 11：ETRP 频率除以 8。
位 11:8	**ETF[3:0]**：外部触发滤波(External trigger filter) 这些位定义了对 ETRP 信号采样的频率和对 ETRP 数字滤波的带宽。实际上，数字滤波器是一个事件计数器，它记录到 N 个事件后会产生一个输出的跳变。 0000：无滤波器，以 f_{DTS} 采样 0001：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=2 0010：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=4 0011：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=8 0100：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=6 0101：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=8 0110：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=6 0111：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=8 1000：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N=6 1001：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N=8 1010：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=5 1011：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=6 1100：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=8 1101：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=5 1110：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=6 1111：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=8

位 7	**MSM**：主/从模式(Master/slave mode) 0：无作用； 1：触发输入(TRGI)上的事件被延迟了，以允许在当前定时器(通过 TRGO)与它的从定时器间的完美同步。这对要求把几个定时器同步到一个单一的外部事件时是非常有用的。

位 6:4	**TS[2:0]**：触发选择(Trigger selection) 这 3 位选择用于同步计数器的触发输入。 000：内部触发 0(ITR0)，TIM1　　100：TI1 的边沿检测器(TI1F_ED) 001：内部触发 1(ITR1)，TIM2　　101：滤波后的定时器输入 1(TI1FP1) 010：内部触发 2(ITR2)，TIM3　　110：滤波后的定时器输入 2(TI2FP2) 011：内部触发 3(ITR3)，TIM4　　111：外部触发输入(ETRF) 关于每个定时器中 ITRx 的细节，参见表 56。 注：这些位只能在未用到(如 SMS=000)时被改变，以避免在改变时产生错误的边沿检测。
位 3	保留，始终读为 0。
位 2:0	**SMS[2:0]**：从模式选择(Slave mode selection) 当选择了外部信号，触发信号(TRGI)的有效边沿与选中的外部输入极性相关(见输入控制寄存器和控制寄存器的说明) 000：关闭从模式– 如果 CEN=1，则预分频器直接由内部时钟驱动。 001：编码器模式 1 – 根据 TI1FP1 的电平，计数器在 TI2FP2 的边沿向上/下计数。 010：编码器模式 2 – 根据 TI2FP2 的电平，计数器在 TI1FP1 的边沿向上/下计数。 011：编码器模式 3 – 根据另一个信号的输入电平，计数器在 TI1FP1 和 TI2FP2 的边沿向上/下数。 100：复位模式 – 选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。 101：门控模式 – 当触发输入(TRGI)为高时，计数器的时钟开启。一旦触发输入变为低，则计数器停止(但不复位)。计数器的启动和停止都是受控的。 110：触发模式 – 计数器在触发输入 TRGI 的上升沿启动(但不复位)，只有计数器的启动是受控的。 111：外部时钟模式 1 – 选中的触发输入(TRGI)的上升沿驱动计数器。 注：如果 TI1F_EN 被选为触发输入(TS=100)时，不要使用门控模式。这是因为，TI1F_ED 在每次 TI1F 变化时输出一个脉冲，然而门控模式是要检查触发输入的电平。

表 56 TIMx 内部触发连接

从定时器	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM2	TIM1	-	TIM3	TIM4
TIM3	TIM1	TIM2	-	TIM4
TIM4	TIM1	TIM2	TIM3	-

1. 如果某个产品中没有相应的定时器，则对应的触发信号 ITRx 也不存在。

13.4.4 DMA/中断使能寄存器(TIMx_DIER)

偏移地址：0x0C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	TDE	保留	CC4DE	CC3DE	CC2DE	CC1DE	UDE	保留	TIE	保留	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

位 15	保留，始终读为 0。
位 14	**TDE**：允许触发 DMA 请求(Trigger DMA request enable) 0：禁止触发 DMA 请求； 1：允许触发 DMA 请求。

位 13	保留，始终读为 0。
位 12	**CC4DE**：允许捕获/比较 4 的 DMA 请求 (Capture/Compare 4 DMA request enable) 0：禁止捕获/比较 4 的 DMA 请求； 1：允许捕获/比较 4 的 DMA 请求。

位 11	**CC3DE**：允许捕获/比较 3 的 DMA 请求 (Capture/Compare 3 DMA request enable) 0：禁止捕获/比较 3 的 DMA 请求； 1：允许捕获/比较 3 的 DMA 请求。
位 10	**CC2DE**：允许捕获/比较 2 的 DMA 请求 (Capture/Compare 2 DMA request enable) 0：禁止捕获/比较 2 的 DMA 请求； 1：允许捕获/比较 2 的 DMA 请求。
位 9	**CC1DE**：允许捕获/比较 1 的 DMA 请求 (Capture/Compare 1 DMA request enable) 0：禁止捕获/比较 1 的 DMA 请求； 1：允许捕获/比较 1 的 DMA 请求。
位 8	**UDE**：允许更新的 DMA 请求(Update DMA request enable) 0：禁止更新的 DMA 请求； 1：允许更新的 DMA 请求。
位 7	保留，始终读为 0。
位 6	**TIE**：触发中断使能(Trigger interrupt enable) 0：禁止触发中断； 1：使能触发中断。
位 5	保留，始终读为 0。
位 4	**CC4IE**：允许捕获/比较 4 中断(Capture/Compare 4 interrupt enable) 0：禁止捕获/比较 4 中断； 1：允许捕获/比较 4 中断。
位 3	**CC3IE**：允许捕获/比较 3 中断(Capture/Compare 3 interrupt enable) 0：禁止捕获/比较 3 中断； 1：允许捕获/比较 3 中断。
位 2	**CC2IE**：允许捕获/比较 2 中断(Capture/Compare 2 interrupt enable) 0：禁止捕获/比较 2 中断； 1：允许捕获/比较 2 中断。
位 1	**CC1IE**：允许捕获/比较 1 中断(Capture/Compare 1 interrupt enable) 0：禁止捕获/比较 1 中断； 1：允许捕获/比较 1 中断。
位 0	**UIE**：允许更新中断(Update interrupt enable) 0：禁止更新中断； 1：允许更新中断。

13.4.5 状态寄存器(TIMx_SR)

偏移地址：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			CC4OF	CC3OF	CC2OF	CC1OF	保留		TIF	保留	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc w0	rc w0	rc w0	rc w0			rc w0		rc w0	rc w0	rc w0	rc w0	rc w0

位 15:13	保留，始终读为 0。
位 12	**CC4OF**：捕获/比较 4 重复捕获标记(Capture/Compare 4 overcapture flag) 参见 CC1OF 描述。
位 11	**CC3OF**：捕获/比较 3 重复捕获标记(Capture/Compare 3 overcapture flag) 参见 CC1OF 描述。
位 10	**CC2OF**：捕获/比较 2 重复捕获标记(Capture/Compare 2 overcapture flag) 参见 CC1OF 描述。
位 9	**CC1OF**：捕获/比较 1 重复捕获标记 (Capture/Compare 1 overcapture flag) 仅当相应的通道被配置为输入捕获时，该标记可由硬件置'1'。写'0'可清除该位。 0： 无重复捕获产生； 1：当计数器的值被捕获到 TIMx_CCR1 寄存器时，CC1IF 的状态已经为'1'。
位 8:7	保留，始终读为 0。
位 6	**TIF**：触发器中断标记(Trigger interrupt flag) 当发生触发事件(当从模式控制器处于除门控模式外的其它模式时，在 TRGI 输入端检测到有效边沿，或门控模式下的任一边沿)时由硬件对该位置'1'。它由软件清'0'。 0：无触发器事件产生； 1：触发器中断等待响应。
位 5	保留，始终读为 0。
位 4	**CC4IF**：捕获/比较 4 中断标记(Capture/Compare 4 interrupt flag) 参考 CC1IF 描述。
位 3	**CC3IF**：捕获/比较 3 中断标记(Capture/Compare 3 interrupt flag) 参考 CC1IF 描述。
位 2	**CC2IF**：捕获/比较 2 中断标记(Capture/Compare 2 interrupt flag) 参考 CC1IF 描述。
位 1	**CC1IF**：捕获/比较 1 中断标记(Capture/Compare 1 interrupt flag) **如果通道 CC1 配置为输出模式：** 当计数器值与比较值匹配时该位由硬件置'1'，但在中心对称模式下除外(参考 TIMx_CR1 寄存器的 CMS 位)。它由软件清'0'。 0：无匹配发生； 1：TIMx_CNT 的值与 TIMx_CCR1 的值匹配。 **如果通道 CC1 配置为输入模式：** 当捕获事件发生时该位由硬件置'1'，它由软件清'0'或通过读 TIMx_CCR1 清'0'。 0：无输入捕获产生； 1：计数器值已被捕获(拷贝)至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)。
位 0	**UIF**：更新中断标记(Update interrupt flag) 当产生更新事件时该位由硬件置'1'。它由软件清'0'。 0：无更新事件产生； 1：更新中断等待响应。当寄存器被更新时该位由硬件置'1'： – 若 TIMx_CR1 寄存器的 UDIS=0、URS=0，当 TIMx_EGR 寄存器的 UG=1 时产生更新(软件对计数器 CNT 重新初始化)；

	– 若 TIMx_CR1 寄存器的 UDIS=0、URS=0，当计数器 CNT 被触发事件重初始化时产生更事件。(参考同步控制寄存器的说明)

13.4.6 事件产生寄存器(TIMx_EGR)

偏移地址:0x14

复位值:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留									TG	保留	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

位 15:7	保留，始终读为 0。
位 6	**TG**：产生触发事件(Trigger generation) 该位由软件置'1'，用于产生一个触发事件，由硬件自动清'0'。 0： 无动作； 1：TIMx_SR 寄存器的 TIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。
位 5	保留，始终读为 0。
位 4	**CC4G**：产生捕获/比较 4 事件(Capture/compare 4 generation) 参考 CC1G 描述。
位 3	**CC3G**：产生捕获/比较 3 事件(Capture/compare 3 generation) 参考 CC1G 描述。
位 2	**CC2G**：产生捕获/比较 2 事件(Capture/compare 2 generation) 参考 CC1G 描述。
位 1	**CC1G**：产生捕获/比较 1 事件(Capture/compare 1 generation) 该位由软件置'1'，用于产生一个捕获/比较事件，由硬件自动清'0'。 0： 无动作； 1：在通道 CC1 上产生一个捕获/比较事件： **若通道 CC1 配置为输出：** 设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。 **若通道 CC1 配置为输入：** 当前的计数器值捕获至 TIMx_CCR1 寄存器；设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。若 CC1IF 已经为 1，则设置 CC1OF=1。
位 0	**UG**：产生更新事件(Update generation) 该位由软件置'1'，由硬件自动清'0'。 0：无动作； 1：重新初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清'0'(但是预分频系 数不变)。若在中心对称模式下或 DIR=0(向上计数)则计数器被清'0'，若 DIR=1(向下计数)则计数器取 TIMx_ARR 的值。

13.4.7 捕获/比较模式寄存器 1(TIMx_CCMR1)

偏移地址：0x18

复位值：0x0000

通道可用于输入(捕获模式)或输出(比较模式)，通道的方向由相应的 CCxS 定义。该寄存器其它位的作用在输入和输出模式下不同。OCxx 描述了通道在输出模式下的功能，ICxx 描述了通道在输出模式下的功能。因此必须注意，同一个位在输出模式和输入模式下的功能是不同的。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

输出比较模式：

位 15	**OC2CE**：输出比较 2 清 0 使能(Output compare 2 clear enable)
位 14:12	**OC2M[2:0]**：输出比较 2 模式(Output compare 2 mode)
位 11	**OC2PE**：输出比较 2 预装载使能(Output compare 2 preload enable)
位 10	**OC2FE**：输出比较 2 快速使能(Output compare 2 fast enable)
位 9:8	**CC2S[1:0]**：捕获/比较 2 选择(Capture/Compare 2 selection) 该位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC2 通道被配置为输出； 01：CC2 通道被配置为输入，IC2 映射在 TI2 上； 10：CC2 通道被配置为输入，IC2 映射在 TI1 上； 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 **注：**CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E='0')才是可写的。
位 7	**OC1CE**：输出比较 1 清 0 使能(Output compare 1 clear enable) 0：OC1REF 不受 ETRF 输入的影响； 1：一旦检测到 ETRF 输入高电平，清除 OC1REF=0。
位 6:4	**OC1M[2:0]**：输出比较 1 模式(Output compare 1 enable) 该 3 位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1 的值。OC1REF 是高有效，而 OC1 的有效电平取决于 CC1P 位。 000：冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用； 001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器(TIMx_CCR1)相同时，强制 OC1REF 为高。 010：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器(TIMx_CCR1)相同时，强制 OC1REF 为低。 011：翻转。当 TIMx_CCR1=TIMx_CNT 时，翻转 OC1REF 的电平。 100：强制为无效电平。强制 OC1REF 为低。 101：强制为有效电平。强制 OC1REF 为高。 110：PWM 模式 1－在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为有效电平，否则为无效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为无效电平(OC1REF=0)，否则为有效电平(OC1REF=1)。 111：PWM 模式 2－在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为无效电平，否则为有效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为有效电平，否则为无效电平。 注：在 PWM 模式 1、2 时，仅在比较结果发生改变或输出比较模式从 Frozen 切换到 PWM 模式时 OCREF 才改变。

位 3	**OC1PE**：输出比较 1 预装载使能(Output compare 1 preload enable) 0：禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器，并且新写入的值立即起作用。 1：开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 预装载值在更新事件到来时被传送至当前寄存器中。 注 1：一旦 LOCK 级别设为 3（TIMx_BDTR 寄存器中的 LOCK 位）并且 CC1S='00'(该通道配置成输出）则该位不能被修改。 注 2：仅在单脉冲模式下(TIMx_CR1 寄存器的 OPM='1')，可以在未确认预装载寄存器情况下使用 PWM 模式，否则其动作不确定。
位 2	**OC1FE**：输出比较 1 快速使能(Output compare 1 fast enable) 该位用于加快 CC 输出对触发器输入事件的响应。 0：根据计数器与 CCR1 的值，CC1 正常操作，即使触发器是打开的。当触发器的输入出现一个有效沿时，激活 CC1 输出的最小延时为 5 个时钟周期。 1：输入到触发器的有效沿的作用就象发生了一次比较匹配。因此，OC 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CC1 输出间的延时被缩短为 3 个时钟周期。 该位只在通道被配置成 PWM1 或 PWM2 模式时起作用。
位 1:0	**CC1S[1:0]**：捕获/比较 1 选择(Capture/Compare 1 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：CC1 通道被配置为输入，IC1 映射在 TI2 上； 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E='0')才是可写的。

输入捕获模式：

位 15:12	**IC2F[3:0]**：输入捕获 2 滤波器(Input capture 2 filter)
位 11:10	**IC2PSC[1:0]**：输入/捕获 2 预分频器(input capture 2 prescaler)

位 9:8	**CC2S[1:0]**：捕获/比较 2 选择 (Capture/compare 2 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC2 通道被配置为输出； 01：CC2 通道被配置为输入，IC2 映射在 TI2 上； 10：CC2 通道被配置为输入，IC2 映射在 TI1 上； 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E='0')才是可写的。
位 7:4	**IC1F[3:0]**：输入捕获 1 滤波器(Input capture 1 filter) 这几位定义了 TI1 输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记录到 N 个事件后会产生一个输出的跳变： 0000：无滤波器，以 f_{DTS} 采样 0001：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=2 0010：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=4 0011：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=8 0100：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=6 0101：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=8 0110：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=6 0111：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=8 1000：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N=6 1001：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N=8 1010：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=5 1011：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=6 1100：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=8 1101：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=5 1110：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=6 1111：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=8 注：在现在的芯片版本中，当 ICxF[3:0]=1、2 或 3 时，公式中的 f_{DTS} 由 CK_INT 替代。

位 3:2	**IC1PSC[1:0]**：输入/捕获 1 预分频器(Input capture 1 prescaler) 这 2 位定义了 CC1 输入(IC1)的预分频系数。 一旦 CC1E='0'(TIMx_CCER 寄存器中)，则预分频器复位。 00：无预分频器，捕获输入口上检测到的每一个边沿都触发一次捕获； 01：每 2 个事件触发一次捕获； 10：每 4 个事件触发一次捕获； 11：每 8 个事件触发一次捕获。
位 1:0	**CC1S[1:0]**：捕获/比较 1 选择 (Capture/Compare 1 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：CC1 通道被配置为输入，IC1 映射在 TI2 上； 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 **注**：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E='0')才是可写的。

13.4.8 捕获/比较模式寄存器 2(TIMx_CCMR2)

偏移地址：0x1C

复位值：0x0000

参看以上 CCMR1 寄存器的描述

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

输出比较模式：

位 15	**OC4CE**：输出比较 4 清 0 使能(Output compare 4 clear enable)
位 14:12	**OC4M[2:0]**：输出比较 4 模式(Output compare 4 mode)
位 11	**OC4PE**：输出比较 4 预装载使能(Output compare 4 preload enable)
位 10	**OC4FE**：输出比较 4 快速使能(Output compare 4 fast enable)
位 9:8	**CC4S**[1:0]：捕获/比较 4 选择(Capture/Compare 4 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC4 通道被配置为输出； 01：CC4 通道被配置为输入，IC4 映射在 TI4 上； 10：CC4 通道被配置为输入，IC4 映射在 TI3 上； 11：CC4 通道被配置为输入，IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中 TIMx_SMCR 寄存器的 TS 位选择)。 **注**：CC4S 仅在通道关闭时(TIMx_CCER 寄存器的 CC4E='0')才是可写的。
位 7	**OC3CE**：输出比较 3 清 0 使能(Output compare 3 clear enable)
位 6:4	**OC3M[2:0]**：输出比较 3 模式(Output compare 3 mode)
位 3	**OC3PE**：输出比较 3 预装载使能(Output compare 3 preload enable)
位 2	**OC3FE**：输出比较 3 快速使能(Output compare 3 fast enable)
位 1:0	**CC3S[1:0]**：捕获/比较 3 选择(Capture/Compare 3 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC3 通道被配置为输出； 01：CC3 通道被配置为输入，IC3 映射在 TI3 上； 10：CC3 通道被配置为输入，IC3 映射在 TI4 上；

	11：CC3 通道被配置为输入，IC3 映射在 TRGI 上。此模式仅工作在内部触发器输入被选 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC3S 仅在通道关闭时(TIMx_CCER 寄存器的 CC3E=’0’)才是可写的。

输入捕获模式：

位 15:12	**IC4F[3:0]**：输入捕获 4 滤波器(Input capture 4 filter)
位 11:10	**IC4PSC[1:0]**：输入/捕获 4 预分频器(input capture 4 prescaler)
位 9:8	**CC4S[1:0]**：捕获/比较 4 选择(Capture/compare 4 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC4 通道被配置为输出； 01：CC4 通道被配置为输入，IC4 映射在 TI4 上； 10：CC4 通道被配置为输入，IC4 映射在 TI3 上； 11：CC4 通道被配置为输入，IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC4S 仅在通道关闭时(TIMx_CCER 寄存器的 CC4E=’0’)才是可写的。
位 7:4	**IC3F[3:0]**：输入捕获 3 滤波器(Input capture 3 filter)
位 3:2	**IC3PSC[1:0]**：输入/捕获 3 预分频器(Input capture 3 prescaler)
位 1:0	**CC3S[1:0]**：捕获/比较 3 选择(Capture/Compare 3 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC3 通道被配置为输出； 01：CC3 通道被配置为输入，IC3 映射在 TI3 上； 10：CC3 通道被配置为输入，IC3 映射在 TI4 上； 11：CC3 通道被配置为输入，IC3 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC3S 仅在通道关闭时(TIMx_CCER 寄存器的 CC3E=’0’)才是可写的。

13.4.9 捕获/比较使能寄存器(TIMx_CCER)

偏移地址：0x20

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		CC4P	CC4E	保留		CC3P	CC3E	保留		CC2P	CC2E	保留		CC1P	CC1E
		rw	rw			rw	rw			rw	rw			rw	rw

位 15:14	保留，始终读为 0。
位 13	**CC4P**：输入/捕获 4 输出极性(Capture/Compare 4 output polarity) 参考 CC1P 的描述。
位 12	**CC4E**：输入/捕获 4 输出使能(Capture/Compare 4 output enable) 参考 CC1E 的描述。
位 11:10	保留，始终读为 0。
位 9	**CC3P**：输入/捕获 3 输出极性(Capture/Compare 3 output polarity) 参考 CC1P 的描述。
位 8	**CC3E**：输入/捕获 3 输出使能(Capture/Compare 3 output enable) 参考 CC1E 的描述。
位 7:6	保留，始终读为 0。
位 5	**CC2P**：输入/捕获 2 输出极性(Capture/Compare 2 output polarity) 参考 CC1P 的描述。

位 4	**CC2E**：输入/捕获 2 输出使能(Capture/Compare 2 output enable) 参考 CC1E 的描述。
位 3:2	保留，始终读为 0。
位 1	**CC1P**：输入/捕获 1 输出极性 (Capture/Compare 1 output polarity) **CC1 通道配置为输出：** 0：OC1 高电平有效 1：OC1 低电平有效 **CC1 通道配置为输入：** 该位选择是 IC1 还是 IC1 的反相信号作为触发或捕获信号。 0：不反相：捕获发生在 IC1 的上升沿；当用作外部触发器时，IC1 不反相。 1：反相：捕获发生在 IC1 的下降沿；当用作外部触发器时，IC1 反相。
位 0	**CC1E**：输入/捕获 1 输出使能(Capture/Compare 1 output enable) **CC1 通道配置为输出：** 0：关闭－OC1 禁止输出。 1：开启－OC1 信号输出到对应的输出引脚。 **CC1 通道配置为输入：** 该位决定了计数器的值是否能捕获入 TIMx_CCR1 寄存器。 0：捕获禁止； 0：捕获使能。

表 57 标准 OCx 通道的输出控制位

CCxE 位	OCx 输出状态
0	禁止输出(OCx=0，OCx_EN=0)
1	OCx = OCxREF + 极性，OCx_EN=1

注：连接到标准 OCx 通道的外部 I/O 引脚状态，取决于 OCx 通道状态和 GPIO 以及 AFIO 寄存器。

13.4.10 计数器(TIMx_CNT)

偏移地址：0x24

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0	**CNT[15:0]**：计数器的值(Counter value)。

13.4.11 预分频器(TIMx_PSC)

偏移地址：0x28

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0	**PSC[15:0]**：预分频器的值(Prescaler value) 计数器的时钟频率 CK_CNT 等于 f_{CK_PSC}/(PSC[15:0]+1)。 PSC 包含了当更新事件产生时装入当前预分频器寄存器的值。

13.4.12 自动重装载寄存器(TIMx_ARR)

偏移地址:0x2C 复位值:0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0	**ARR[15:0]**: 自动重装载的值(Auto reload value) ARR 包含了将要传送至实际的自动重装载寄存器的数值。 详细参考章节有关 ARR 的更新和动作。 当自动重装载的值为空时，计数器不工作。

13.4.13 捕获/比较寄存器 1(TIMx_CCR1)

偏移地址：0x34 复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
Rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

位 15:0	**CCR1[15:0]**: 捕获/比较 1 的值(Capture/Compare 1 value) **若 CC1 通道配置为输出：** CCR1 包含了装入当前捕获/比较 1 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC1PE 位)中未选择预装载特性，写入的数值会被立即传输至当前 寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 1 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC1 端口上产生输出信号。 **若 CC1 通道配置为输入：** CCR1 包含了由上一次输入捕获 1 事件(IC1)传输的计数器值。CCR1 是只读寄存器。

13.4.14 捕获/比较寄存器 2(TIMx_CCR2)

偏移地址：0x38 复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

位 15:0	**CCR2[15:0]**: 捕获/比较 2 的值(Capture/Compare 2 value) **若 CC2 通道配置为输出：** CCR2 包含了装入当前捕获/比较 2 寄存器的值(预装载值)。 如果在 TIMx_CCMR2 寄存器(OC2PE 位)中未选择预装载特性，写入的数值会被立即传输至当前 寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 2 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC2 端口上产生输出信号。 **若 CC2 通道配置为输入：** CCR2 包含了由上一次输入捕获 2 事件(IC2)传输的计数器值。CCR2 是只读寄存器

13.4.15 捕获/比较寄存器 3(TIMx_CCR3)

偏移地址：0x3C 复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

位 15:0	**CCR3[15:0]**: 捕获/比较 3 的值(Capture/Compare 3 value) **若 CC3 通道配置为输出：** CCR3 包含了装入当前捕获/比较 3 寄存器的值(预装载值)。 如果在 TIMx_CCMR3 寄存器(OC3PE 位)中未选择预装载特性，写入的数值会被立即传输至当前 寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 3 寄存器中。当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC3 端口上产生输出信号。 **若 CC3 通道配置为输入：** CCR3 包含了由上一次输入捕获 3 事件(IC3)传输的计数器值。CCR3 是只读寄存器

13.4.16 捕获/比较寄存器 4(TIMx_CCR4)

偏移地址：0x40 复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

位 15:0	**CCR4[15:0]**: 捕获/比较 4 的值 (Capture/Compare 4 value) **若 CC4 通道配置为输出：** CCR4 包含了装入当前捕获/比较 4 寄存器的值(预装载值)。 如果在 TIMx_CCMR4 寄存器(OC4PE 位)中未选择预装载特性，写入的数值会被立即传输至当前 寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 4 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC4 端口上产生输出信号。 **若 CC4 通道配置为输入：** CCR4 包含了由上一次输入捕获 4 事件(IC4)传输的计数器值。CCR4 是只读寄存器

13.4.17 DMA 控制寄存器(TIMx_DCR)

偏移地址：0x4C 复位值：0x0000

保留	DBL[4:0]	保留	DBA[4:0]
	rw rw rw rw rw		rw rw rw rw rw

位 15:13	保留，始终读为 0。
位 12:8	**DBL[4:0]**: DMA 连续传送长度(DMA burst length) 这些位定义了 DMA 在连续模式下的传送长度(当对 TIMx_DMAR 寄存器进行读或写时，定时器则进行一次连续传送)，即：定义传输的字节数目： 00000：1 个字节　　00001：2 个字节 00010：3 个字节　　..... .....　　10001：18 个字节
位 7:5	保留，始终读为 0。
位 4:0	**DBA[4:0]**: DMA 基地址(DMA base address) 这些位定义了 DMA 在连续模式下的基地址(当对 TIMx_DMAR 寄存器进行读或写时)，DBA 定义为从 TIMx_CR1 寄存器所在地址开始的偏移量： 00000：TIMx_CR1， 00001：TIMx_CR2， 00010：TIMx_SMCR， .....

13.4.18 连续模式的 DMA 地址(TIMx_DMAR)

偏移地址：0x4C 复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0	**DMAB[15:0]**: DMA 连续传送寄存器(DMA register for burst accesses) 对 TIMx_DMAR 寄存器的读或写会导致对以下地址所在寄存器的存取操作： （TIMx_CR1 地址）+（ DBA + DMA 索引）X4，其中： “TIMx_CR1 地址”是控制寄存器 1(TIMx_CR1)所在的地址； “DBA”是 TIMx_DCR 寄存器中定义的基地址； “DMA 索引”是由 DMA 自动控制的偏移量，它取决于 TIMx_DCR 寄存器中定义的 DBL。

13.4.19 TIMx 寄存器图

下表中将TIMx的所有寄存器映射到一个16位可寻址(编址)空间。

表 58 TIMx – 寄存器图和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	TIMx_CR1	保留																						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
	复位值																							0	0	0	0	0	0	0	0	0	0
004h	TIMx_CR2	保留																								TI1S	MMS[2:0]			CCDS	保留		
	复位值																									0	0	0	0	0			
008h	TIMx_SMCR	保留																0	0	ETPS[1:0]		EFT[3:0]				0	TS[2:0]				SMS[2:0]		
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
00Ch	TIMx_DIER	保留																	0		0	0	0	0	0		0		0	0	0	0	0
	复位值																																
010h	TIMx_SR	保留																			0	0	0	0	保留		0		0	0	0	0	0
	复位值																																
014h	TIMx_EGR	保留																									0		0	0	0	0	0
	复位值																																
018h	TIMx_CCMR1 输出比较模式	保留																0	0	OC2M[2:0]				CC2S[1:0]		0	OC1M[2:0]				CC1S[1:0]		
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
	TIMx_CCMR1 输入捕获模式	保留																IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01Ch	TIMx_CCMR2 输出比较模式	保留																0	0	OC4M[2:0]				CC4S[1:0]		0	OC3M[2:0]				CC3S[1:0]		
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
	TIMx_CCMR2 输入捕获模式	保留																IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	TIMx_CCER	保留																			0	0	保留	0	0	保留		0	0	保留		0	0
	复位值																																
024h	TIMx_CNT	保留																CNT[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028h	TIMx_PSC	保留																PSC[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
02Ch	TIMx_ARR	保留																ARR[15:0]															
	复位值																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
030h	保留																																
034h	TIMx_CCR1	保留																CCR1[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
038h	TIMx_CCR2	保留																CCR2[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03Ch	TIMx_CCR3	保留																CCR3[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
040h	TIMx_CCR4	保留																CCR4[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
044h	保留																																
048h	TIMx_DCR	保留																保留			DBL[4:0]					保留			DBA[4:0]			
	复位值																				0	0	0	0	0				0	0	0	0	0
04Ch	TIMx_DMAR	保留																DMAB[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器的起始地址，参见表 1。

14 实时时钟（RTC）

14.1 RTC 简介

实时时钟是一个独立的定时器。RTC 模块拥有一组连续计数的计数器，在相应软件配置下，可提供时钟日历的功能。修改计数器的值可以重新设置系统当前的时间和日期。

RTC 模块和时钟配置系统(RCC_BDCR 寄存器)处于后备区域，即在系统复位或从待机模式唤醒后，RTC 的设置和时间维持不变。

系统复位后，对后备寄存器和 RTC 的访问被禁止，这是为了防止对后备区域(BKP)的意外写操作。执行以下操作将使能对后备寄存器和 RTC 的访问：

- 设置寄存器 RCC_APB1ENR 的 PWREN 和 BKPEN 位，使能电源和后备接口时钟
- 设置寄存器 PWR_CR 的 DBP 位，使能对后备寄存器和 RTC 的访问。

14.2 主要特性

- 可编程的预分频系数：分频系数最高为 2^{20}。
- 32 位的可编程计数器，可用于较长时间段的测量。
- 2 个分离的时钟：用于 APB1 接口的 PCLK1 和 RTC 时钟(RTC 时钟的频率必须小于 PCLK1 时钟 频率的四分之一以上)。
- 可以选择以下三种 RTC 的时钟源：
  - HSE 时钟除以 128；
  - LSE 振荡器时钟；
  - LSI 振荡器时钟(详见 8.2.8 节 RTC 时钟)。
- 2 个独立的复位类型：
  - APB1 接口由系统复位；
  - RTC 核心(预分频器、闹钟、计数器和分频器)只能由后备域复位(详见 8.1.3 节)。
- 3 个专门的可屏蔽中断：
  - 闹钟中断，用来产生一个软件可编程的闹钟中断。
  - 秒中断，用来产生一个可编程的周期性中断信号(最长可达 1 秒)。
  - 溢出中断，指示内部可编程计数器溢出并回转为 0 的状态。

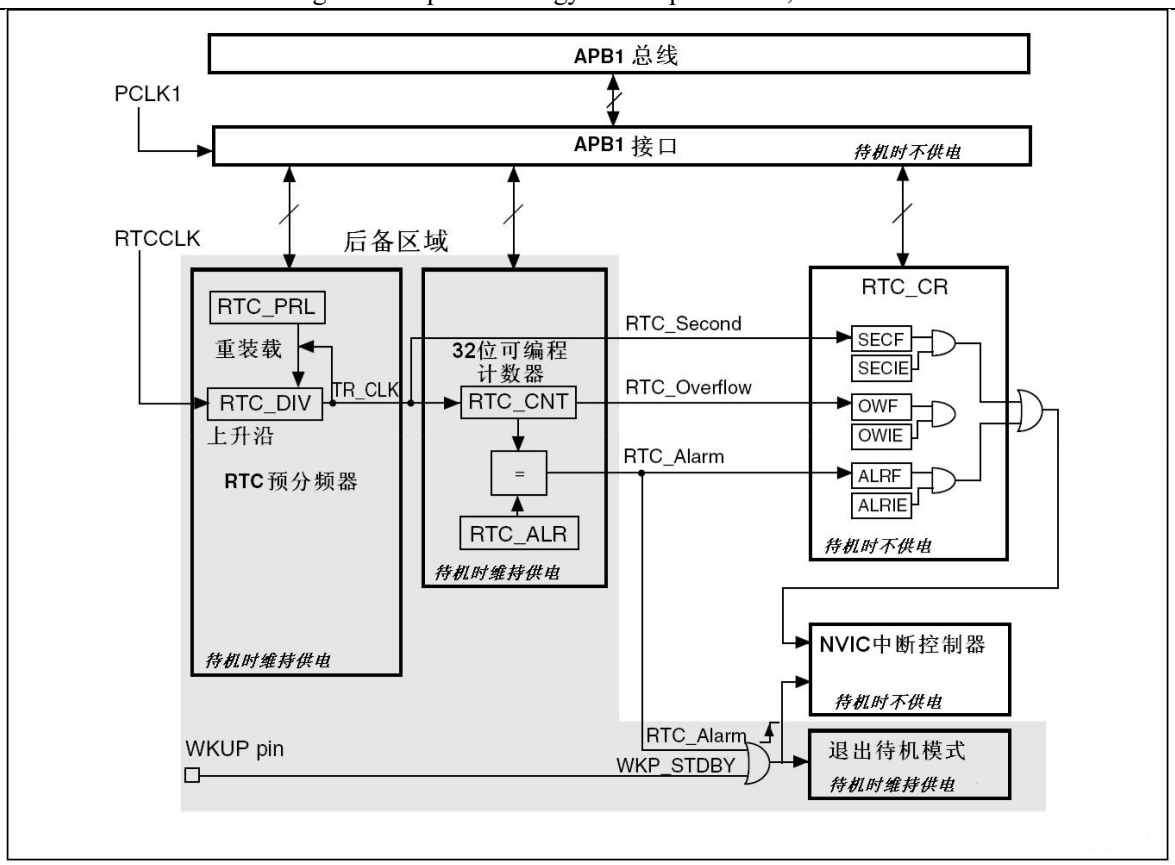
14.3 功能描述

14.3.1 概述

RTC 由两个主要部分组成(参见下图)。第一部分(APB1 接口)用来和 APB1 总线相连。此单元还包含一组 16 位寄存器，可通过 APB1 总线对其进行读写操作(参见 15.4 节)。APB1 接口由 APB1 总线 时钟驱动，用来与 APB1 总线接口。

另一部分(RTC 核心)由一组可编程计数器组成，分成两个主要模块。第一个模块是 RTC 的预分频模块，它可编程产生最长为 1 秒的 RTC 时间基准 TR_CLK。RTC 的预分频模块包含了一个 20 位的可编程分频器(RTC 预分频器)。如果在 RTC_CR 寄存器中设置了相应的允许位，则在每个 TR_CLK 周期中 RTC 产生一个中断(秒中断)。第二个模块是一个 32 位的可编程计数器，可被初始化为当前的系统时间。系统时间按 TR_CLK 周期累加并与存储在 RTC_ALR 寄存器中的可编程时 间相比较，如果 RTC_CR 控制寄存器中设置了相应允许位，比较匹配时将产生一个闹钟中断。

图 131 简化的 RTC 框图



14.3.2 复位过程

除了 RTC_PRL、RTC_ALR、RTC_CNT 和 RTC_DIV 寄存器外，所有的系统寄存器都由系统复位或电源复位进行异步复位。

RTC_PRL、RTC_ALR、RTC_CNT 和 RTC_DIV 寄存器仅能通过备份域复位信号复位，详见第8.1.3 节。

14.3.3 读 RTC 寄存器

RTC 核完全独立于 RTC APB1 接口。软件通过 APB1 接口访问 RTC 的预分频值、计数器值和闹钟值。但是，相关的可读寄存器只在与 RTC APB1 时钟进行重新同步的 RTC 时钟的上升沿被更新。RTC 标志也是如此的。

这意味着，如果 APB1 接口曾经被关闭，而读操作又是在刚刚重新开启 APB1 之后，则在第一次的内部寄存器更新之前，从 APB1 上读出的 RTC 寄存器数值可能被破坏了(通常读到 0)。下述几种 情况下能够发生这种情形：

- 发生系统复位或电源复位
- 系统刚从待机模式唤醒(参见第 6.3 节：低功耗模式)。
- 系统刚从停机模式唤醒(参见第 6.3 节：低功耗模式)。 所有以上情况中，APB1 接口被禁止时(复位、无时钟或断电)RTC 核仍保持运行状态。

因此，若在读取 RTC 寄存器时，RTC 的 APB1 接口曾经处于禁止状态，则软件首先必须等待 RTC_CRL 寄存器中的 RSF 位(寄存器同步标志)被硬件置'1'。

注： *RTC 的 APB1 接口不受 WFI 和 WFE 等低功耗模式的影响。*

14.3.4 配置 RTC 寄存器

必须设置RTC_CRL 寄存器中的CNF 位，使RTC 进入配置模式后，才能写入RTC_PRL、RTC_CNT、RTC_ALR 寄存器。

另外，对RTC任何寄存器的写操作，都必须在前一次写操作结束后进行。可以通过查询RTC_CR寄存器中的RTOFF状态位，判断RTC寄存器是否处于更新中。仅当RTOFF状态位是'1'时，才可以写入RTC寄存器。

配置过程：

1. 查询RTOFF位，直到RTOFF的值变为'1'
2. 置CNF值为1，进入配置模式
3. 对一个或多个RTC寄存器进行写操作
4. 清除CNF标志位，退出配置模式
5. 查询RTOFF，直至RTOFF位变为'1'以确认写操作已经完成。

仅当CNF标志位被清除时，写操作才能进行，这个过程至少需要3个RTCCLK周期。

14.3.5 RTC标志的设置

在每一个RTC核心的时钟周期中，更改RTC计数器之前设置RTC秒标志(SECF)。 在计数器到达0x0000之前的最后一个RTC时钟周期中，设置RTC溢出标志(OWF)。

在计数器的值到达闹钟寄存器的值加1(RTC_ALR+1)之前的RTC时钟周期中，设置RTC_Alarm和RTC闹钟标志(ALRF)。对RTC闹钟的写操作必须使用下述过程之一与RTC秒标志同步：

- 使用RTC闹钟中断，并在中断处理程序中修改RTC闹钟和/或RTC计数器。
- 等待RTC控制寄存器中的SECF位被设置，再更改RTC闹钟和/或RTC计数器。

图 132 RTC秒和闹钟波形图示例，PR=0003，ALARM=00004

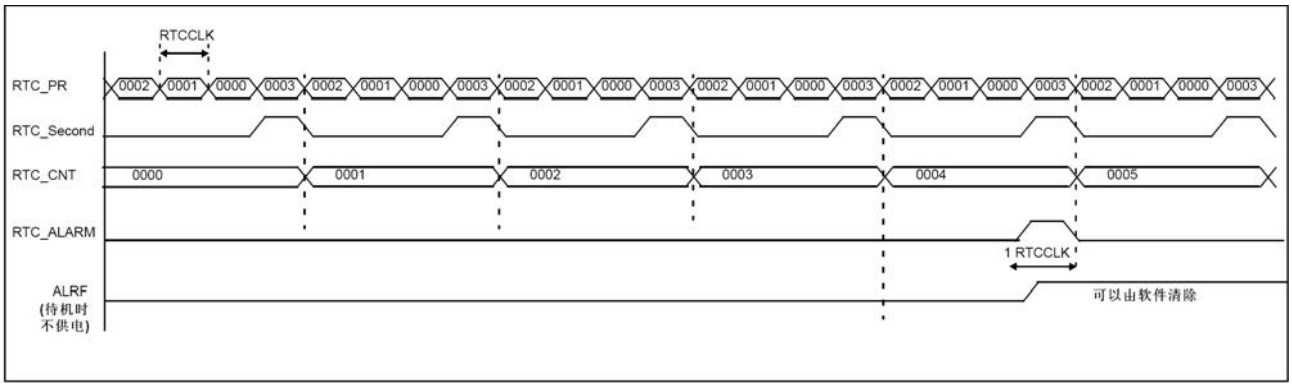
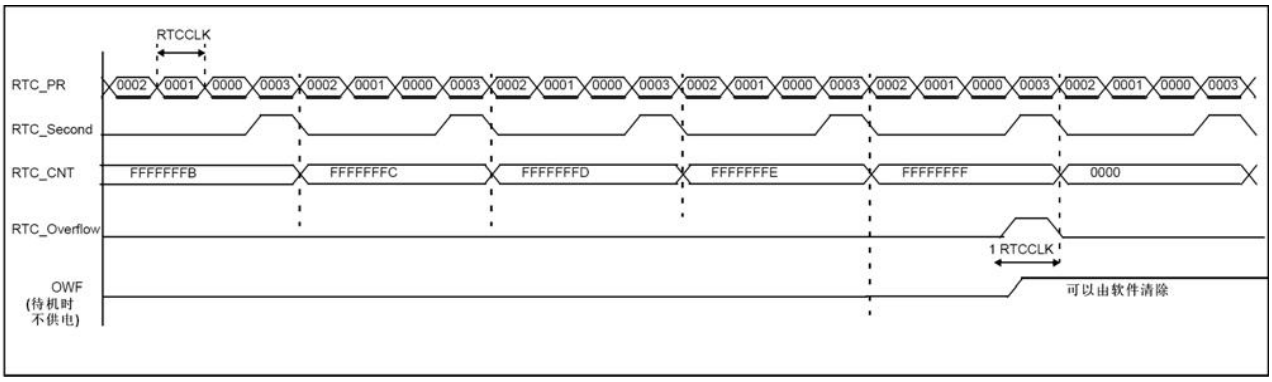


图 133 RTC溢出波形图示例，PR=0003



14.4 RTC 寄存器描述

关于寄存器描述中的缩略词，请参考 2.1 节。 可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

14.4.1 RTC 控制寄存器高位(RTC_CRH)

地址偏移量：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留													OWIE	ALRIE	SECIE
													rw	rw	rw

位 15:3	保留，被硬件强制为 0。
位 2	**OWIE**：允许溢出中断位(Overflow interrupt enable) 0：屏蔽(不允许)溢出中断 1：允许溢出中断
位 1	**ALRIE**：允许闹钟中断(Alarm interrupt enable) 0：屏蔽(不允许)闹钟中断 1：允许闹钟中断
位 0	**SECIE**：允许秒中断(Second interrupt enable) 0：屏蔽(不允许)秒中断 1：允许秒中断

这些位用来屏蔽中断请求。注意：系统复位后所有的中断被屏蔽，因此可通过写 RTC 寄存器来确保在初始化后没有挂起的中断请求。当外设正在完成前一次写操作时(标志位 RTOFF=0)，不能对 RTC_CRH 寄存器进行写操作。

RTC 功能由这个控制寄存器控制。一些位的写操作必须经过一个特殊的配置过程来完成(见 15.3.4 节)。

14.4.2 RTC 控制寄存器低位(RTC_CRL)

偏移地址：0x04

复位值：0x0020

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										RTOFF	CNF	RSF	OWF	ALRF	SECF
										r	rw	rc_w0	rc_w0	rc_w0	rc_w0

位 15:6	保留，被硬件强制为 0。
位 5	**RTOFF**：RTC 操作关闭(RTC operation OFF) RTC 模块利用这位来指示对其寄存器进行的最后一次操作的状态，指示操作是否完成。若此位为'0'，则表示无法对任何的 RTC 寄存器进行写操作。此位为只读位。 0：上一次对 RTC 寄存器的写操作仍在进行; 1：上一次对 RTC 寄存器的写操作已经完成。
位 4	**CNF**：配置标志(Configuration flag) 此位必须由软件置'1'以进入配置模式，从而允许向 RTC_CNT、RTC_ALR 或 RTC_PRL 寄存写入数据。只有当此位在被置'1'并重新由软件清'0'后，才会执行写操作。 0：退出配置模式(开始更新 RTC 寄存器); 1：进入配置模式。

位 3	**RSF**：寄存器同步标志(Registers synchronized flag) 每当 RTC_CNT 寄存器和 RTC_DIV 寄存器由软件更新或清'0'时，此位由硬件置'1'。在 APB1 复位后，或 APB1 时钟停止后，此位必须由软件清'0'。要进行任何的读操作之前，用户程序必须等待这位被硬件置'1'，以确保 RTC_CNT、RTC_ALR 或 RTC_PRL 已经被同步。 0：寄存器尚未被同步； 1：寄存器已经被同步。
位 2	**OWF**：溢出标志(Overflow flag) 当 32 位可编程计数器溢出时，此位由硬件置'1'。如果 RTC_CRH 寄存器中 OWIE=1，则产生断。此位只能由软件清'0'。对此位写'1'是无效的。 0：无溢出； 1：32 位可编程计数器溢出。
位 1	**ALRF**：闹钟标志(Alarm flag) 当 32 位可编程计数器达到 RTC_ALR 寄存器所设置的预定值，此位由硬件置'1'。如果 RTC_CRH 寄存器中 ALRIE=1，则产生中断。此位只能由软件清'0'。对此位写'1'是无效的。 0：无闹钟； 1：有闹钟。
位 0	**SECF**：秒标志(Second flag) 当 32 位可编程预分频器溢出时，此位由硬件置'1'同时 RTC 计数器加 1。因此，此标志为分辨率编程的 RTC 计数器提供一个周期性的信号(通常为 1 秒)。如果 RTC_CRH 寄存器中 SECIE=1，产生中断。此位只能由软件清除。对此位写'1'是无效的。 0：秒标志条件不成立； 1：秒标志条件成立。

RTC 的功能由这个控制寄存器控制。当前一个写操作还未完成时(RTOFF=0 时，详见 15.3.4 节)，不能写 RTC_CR 寄存器。

注： 1 任何标志位都将保持挂起状态，直到适当的 RTC_CR 请求位被软件复位，表示所请求的中断已经被接受。

2 在复位时禁止所有中断，无挂起的中断请求，可以对 RTC 寄存器进行写操作。

3 当 APB1 时钟不运行时，OWF、ALRF、SECF 和 RSF 位不被更新。

4 OWF、ALRF、SECF 和 RSF 位只能由硬件置位，由软件来清零。

5 若 ALRF=1 且 ALRIE=1，则允许产生 RTC 全局中断。如果在 EXTI 控制器中允许产生 EXTI 线 17 中断，则允许产生 RTC 全局中断和 RTC 闹钟中断。

6 若 ALRF=1，如果在 EXTI 控制器中设置了 EXTI 线 17 的中断模式，则允许产生 RTC 闹钟中断；如果在 EXTI 控制器中设置了 EXTI 线 17 的事件模式，则这条线上会产生一个脉冲(不会产生 RTC 闹钟中断)。

14.4.3 RTC 预分频装载寄存器(RTC_PRLH/RTC_PRLL)

预分频装载寄存器用来保存 RTC 预分频器的周期计数值。它们受 RTC_CR 寄存器的 RTOFF 位保护，仅当 RTOFF 值为'1'时允许进行写操作。

RTC 预分频装载寄存器高位(RTC_PRLH)

偏移地址：0x08

只写(参见 15.3.4 节)

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												PRL[19:16]			
												w	w	w	w

位 15:6	保留，被硬件强制为 0。
位 3:0	**PRL[19:16]**：RTC 预分频装载值高位(RTC prescaler reload value high) 根据以下公式，这些位用来定义计数器的时钟频率： $f_{TR\_CLK} = f_{RTCCLK}/(PRL[19:0]+1)$ 注：不推荐使用 0 值，否则无法正确的产生 RTC 中断和标志位。

RTC 预分频装载寄存器低位(RTC_PRLL)

偏移地址：0x0C

只写(参见 15.3.4 节) 复位值：0x8000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRL[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 15:0	**PRL[15:0]**：RTC 预分频装载值低位 根据以下公式，这些位用来定义计数器的时钟频率： $f_{TR\_CLK} = f_{RTCCLK}/(PRL[19:0]+1)$

注：　如果输入时钟频率是 32.768kHz(f_{RTCCLK})，这个寄存器中写入 7FFFh 可获得周期为 1 秒钟的信号。

14.4.4 RTC 预分频器余数寄存器(RTC_DIVH / RTC_DIVL)

在 TR_CLK 的每个周期里，RTC 预分频器中计数器的值都会被重新设置为 RTC_PRL 寄存器的值。用户可通过读取 RTC_DIV 寄存器，以获得预分频计数器的当前值，而不停止分频计数器的工作，从而获得精确的时间测量。此寄存器是只读寄存器，其值在 RTC_PRL 或 RTC_CNT 寄存器中的值发生改变后，由硬件重新装载。

RTC 预分频器余数寄存器高位(RTC_DIVH)

偏移地址：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												RTC_DIV[19:16]			
												r	r	r	r

位 15:4	保留
位 3:0	**RTC_DIV[19:16]**：RTC 时钟分频器余数高位(RTC clock divider high)

RTC 预分频器余数寄存器低位(RTC_DIVL)

偏移地址：0x14

复位值：0x8000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_DIV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 15:0	**RTC_DIV[15:0]**：RTC 时钟器余数低位(RTC clock divider low)

14.4.5 RTC 计数器寄存器 (RTC_CNTH / RTC_CNTL)

RTC 核有一个 32 位可编程的计数器，可通过两个 16 位的寄存器访问。计数器以预分频器产生的TR_CLK时间基准为参考进行计数。RTC_CNT寄存器用来存放计数器的计数值。他们受RTC_CR的位RTOFF写保护，仅当RTOFF值为'1'时，允许写操作。在高或低寄存器(RTC_CNTH或RTC_CNTL)上的写操作，能够直接装载到相应的可编程计数器，并且重新装载 RTC 预分频器。当进行读操作时，直接返回计数器内的计数值(系统时间)。

RTC 计数器寄存器高位(RTC_CNTH)

偏移地址：0x18

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_CNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0	**RTC_CNT[31:16]**：RTC 计数器高位(RTC counter high) 可通过读 RTC_CNTH 寄存器来获得 RTC 计数器当前值的高位部分。要对此寄存器进行写操作 前，必须先进入配置模式(参见 15.3.4 节)。

RTC 计数器寄存器低位(RTC_CNTL)

偏移地址：0x1C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0	**RTC_CNT[15:0]**：RTC 计数器低位。 可通过读 RTC_CNTL 寄存器来获得 RTC 计数器当前值的低位部分。要对此寄存器进行写操作，必须先进入配置模式(参见 15.3.4 节)。

14.4.6 RTC 闹钟寄存器(RTC_ALRH/RTC_ALRL)

当可编程计数器的值与 RTC_ALR 中的 32 位值相等时，即触发一个闹钟事件，并且产生 RTC 闹钟中断。此寄存器受 RTC_CR 寄存器里的 RTOFF 位写保护，仅当 RTOFF 值为'1'时，允许写操作。

RTC 闹钟寄存器高位(RTC_ALRH)

偏移地址：0x20

只写(参见 15.3.4 节)

复位值：0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_ALR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 15:0	**RTC_ALR[31:16]**：RTC 闹钟值高位(RTC alarm high) 此寄存器用来保存由软件写入的闹钟时间的高位部分。要对此寄存器进行写操作，必须先进入配置模式(参见 15.3.4 节)。

RTC 闹钟寄存器低位(RTC_ALRL)

偏移地址：0x24

只写(参见 15.3.4 节)

复位值：0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_ALR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 15:0	**RTC_ALR[15:0]**：RTC 闹钟值低位(RTC alarm low) 此寄存器用来保存由软件写入的闹钟时间的低位部分。要对此寄存器进行写操作，必须先进入配置模式(参见 15.3.4 节)。

14.4.7 RTC 寄存器映像

RTC 寄存器是 16 位可寻址寄存器，具体描述如下：

表 59 RTC-寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	RTC_CRH	保留																													OWIE	ALRIE	SECIE
	复位值																														0	0	0
004h	RTC_CRL	保留																										RTOFF	CNF	RSF	OWF	ALRF	SECF
	复位值																											0	0	0	0	0	0
008h	RTC_PRLH	保留																													PRL[19:16]		
	复位值																														0	0	0	0
00Ch	RTC_PRLL	保留																PRL[15:0]															
	复位值																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	RTC_DIVH	保留																DIV[31:16]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	RTC_DIVL	保留																DIV[15:0]															
	复位值																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	RTC_CNTH	保留																CNT[31:16]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01Ch	RTC_CNTL	保留																CNT[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	RTC_ALRH	保留																ALR[31:16]															
	复位值																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
024h	RTC_ALRL	保留																ALR[15:0]															
	复位值																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

有关寄存器的起始地址，参见表 1。

15 独立看门狗（IWDG）

15.1 简介

HK32F10xxx 内置两个看门狗，提供了更高的安全性、时间的精确性和使用的灵活性。两个看门狗设备(独立看门狗和窗口看门狗)可用来检测和解决由软件错误引起的故障；当计数器达到给定的超时值时，触发一个中断(仅适用于窗口型看门狗)或产生系统复位。

独立看门狗(IWDG)由专用的低速时钟(LSI)驱动，即使主时钟发生故障它也仍然有效。窗口看门狗由从 APB1 时钟分频后得到的时钟驱动，通过可配置的时间窗口来检测应用程序非正常的过迟或过早的操作。

IWDG 最适合应用于那些需要看门狗作为一个在主程序之外，能够完全独立工作，并且对时间精度要求较低的场合。WWDG 最适合那些要求看门狗在精确计时窗口起作用的应用程序。
关于窗口看门狗的详情，请参看第 20 章。

15.2 IWDG 主要性能

- 自由运行的递减计数器
- 时钟由独立的 RC 振荡器提供(可在停止和待机模式下工作)
- 看门狗被激活后，则在计数器计数至 0x000 时产生复位

15.3 IWDG 功能描述

图 134 为独立看门狗模块的功能框图。

在键寄存器(IWDG_KR)中写入 0xCCCC，开始启用独立看门狗；此时计数器开始从其复位值 0xFFF 递减计数。当计数器计数到末尾 0x000 时，会产生一个复位信号(IWDG_RESET)。

无论何时，只要在键寄存器 IWDG_KR 中写入 0xAAAA， IWDG_RLR 中的值就会被重新加载到计数器，从而避免产生看门狗复位。

15.3.1 硬件看门狗

如果用户在选择字节中启用了“硬件看门狗”功能，在系统上电复位后，看门狗会自动开始运行；如果在计数器计数结束前，若软件没有向键寄存器写入相应的值，则系统会产生复位。

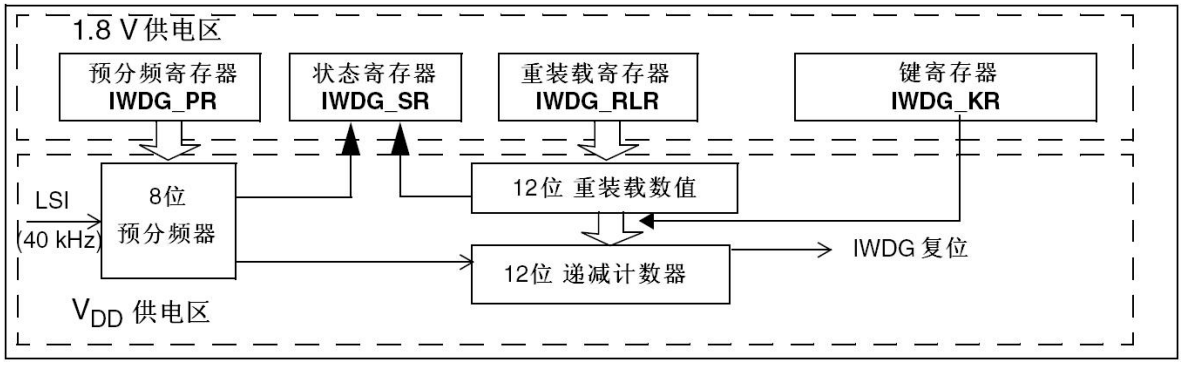
15.3.2 寄存器访问保护

IWDG_PR 和IWDG_RLR 寄存器具有写保护功能。要修改这两个寄存器的值，必须先向 IWDG_KR 寄存器中写入 0x5555。以不同的值写入这个寄存器将会打乱操作顺序，寄存器将重新被保护。重装载操作(即写入 0xAAAA)也会启动写保护功能。
状态寄存器指示预分频值和递减计数器是否正在被更新。

15.3.3 调试模式

当微控制器进入调试模式时(Cortex-M3 核心停止)，根据调试模块中的 DBG_IWDG_STOP 配置位的状态，IWDG 的计数器能够继续工作或停止。详见有关调试模块的章节。

图 134 独立看门狗框图



注：　看门狗功能处于 VDD 供电区，即在停机和待机模式时仍能正常工作。

表 60 看门狗超时时间(40kHz 的输入时钟(LSI))

预分频系数	PR[2:0]位	最短时间(ms) RL[11:0] = 0x000	最长时间(ms) RL[11:0] = 0xFFF
/4	0	0.1	409.6
/8	1	0.2	819.2
/16	2	0.4	1638.4
/32	3	0.8	3276.8
/64	4	1.6	6553.6
/128	5	3.2	13107.2
/256	(6 或 7)	6.4	26214.4

注：　这些时间是按照 40kHz 时钟给出。实际上，MCU 内部的 RC 频率会在 30kHz 到 60kHz 之间变化。 此外，即使 RC 振荡器的频率是精确的，确切的时序仍然依赖于 APB 接口时钟与 RC 振荡器时钟 之间的相位差，因此总会有一个完整的 RC 周期是不确定的。

通过对 LSI 进行校准可获得相对精确的看门狗超时时间。有关 LSI 校准的问题，详见 8.2.5 节。

15.4 IWDG 寄存器描述

关于在寄存器描述里面所用到的缩写，详见第 2.1 节。 可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

15.4.1 键寄存器(IWDG_KR)

地址偏移：0x00

复位值：0x0000 0000 (在待机模式复位)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 31:16	保留，始终读为 0。
位 15:0	**KEY[15:0]**: 键值(只写寄存器，读出值为 0x0000) (Key value) 软件必须以一定的间隔写入 0xAAAA，否则，当计数器为 0 时，看门狗会产生复位。 写入 0x5555 表示允许访问 IWDG_PR 和 IWDG_RLR 寄存器。(见 16.3.2 节) 写入 0xCCCC，启动看门狗工作(若选择了硬件看门狗则不受此命令字限制)。

15.4.2 预分频寄存器(IWDG_PR)

地址偏移：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留													PR[2:0]		
													rw	rw	rw

位 31:3	保留，始终读为 0。
位 2:0	**PR[2:0]**: 预分频因子(Prescaler divider) 这些位具有写保护设置，参见 16.3.2 节。通过设置这些位来选择计数器时钟的预分频因子。要改变预分频因子，IWDG_SR 寄存器的 PVU 位必须为 0。 000: 预分频因子=4　　100: 预分频因子=64 001: 预分频因子=8　　101: 预分频因子=128 010: 预分频因子=16　　110: 预分频因子=256 011: 预分频因子=32　　111: 预分频因子=256 注意：对此寄存器进行读操作，将从 VDD 电压域返回预分频值。如果写操作正在进行，则读回的值可能是无效的。因此，只有当 IWDG_SR 寄存器的 PVU 位为 0 时，读出的值才有效。

15.4.3 重装载寄存器(IWDG_RLR)

地址偏移：0x08

复位值：0x0000 0FFF(待机模式时复位)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				RL[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:12	保留，始终读为 0。
位 11:0	**RL[11:0]**: 看门狗计数器重装载值(Watchdog counter reload value) 这些位具有写保护功能，参看 16.3.2 节。用于定义看门狗计数器的重装载值，每当向 IWDG_KR 寄存器写入 0xAAAA 时，重装载值会被传送到计数器中。随后计数器从这个值开始递减计数。看门狗超时周期可通过此重装载值和时钟预分频值来计算，参照表 60。 只有当 IWDG_SR 寄存器中的 RVU 位为 0 时，才能对此寄存器进行修改。 注：对此寄存器进行读操作，将从 VDD 电压域返回预分频值。如果写操作正在进行，则读回的值可能是无效的。因此，只有当 IWDG_SR 寄存器的 RVU 位为 0 时，读出的值才有效。

15.4.4 状态寄存器（IWDG_SR）

地址偏移：0x0C

复位值：0x0000 0000 (待机模式时不复位)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														RVU	PVU
														r	r

位 31:2	保留。
位 1	**RVU**: 看门狗计数器重装载值更新(Watchdog counter reload value update) 此位由硬件置'1'用来指示重装载值的更新正在进行中。当在 VDD 域中的重装载更新结束后，此位由硬件清'0'(最多需 5 个 40kHz 的 RC 周期)。重装载值只有在 RVU 位被清'0'后才可更新。
位 0	**PVU**: 看门狗预分频值更新(Watchdog prescaler value update) 此位由硬件置'1'用来指示预分频值的更新正在进行中。当在 VDD 域中的预分频值更新结束后，此位由硬件清'0'(最多需 5 个 40kHz 的 RC 周期)。预分频值只有在 PVU 位被清'0'后才可更新。

注：　如果在应用程序中使用了多个重装载值或预分频值，则必须在 RVU 位被清除后才能重新改变预 装载值，在 PVU 位被清除后才能重新改变预分频值。然而，在预分频和/或重装值更新后，不 必等待 RVU 或 PVU 复位，可继续执行下面的代码。(即是在低功耗模式下，此写操作仍会被继 续执行完成。)

15.4.5 IWDG 寄存器映像

表 61 IWDG 寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	IWDG_KR	保留																KEY[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004h	IWDG_PR	保留																														PR[2:0]	
	复位值																													0	0	0	
008h	IWDG_RLR	保留																				RL[11:0]											
	复位值																					1	1	1	1	1	1	1	1	1	1	1	1
00Ch	IWDG_SR	保留																														RVU	PVU
	复位值																															0	0

有关寄存器的起始地址，参见表 1。

16 窗口看门狗（WWDG）

16.1 WWDG 简介

窗口看门狗通常被用来监测，由外部干扰或不可预见的逻辑条件造成的应用程序背离正常的运行序列而产生的软件故障。除非递减计数器的值在 T6 位变成 0 前被刷新，看门狗电路在达到预置的时间周期时，会产生一个 MCU 复位。在递减计数器达到窗口寄存器数值之前，如果 7 位的递减 计数器数值(在控制寄存器中)被刷新， 那么也将产生一个 MCU 复位。这表明递减计数器需要在一个有限的时间窗口中被刷新。

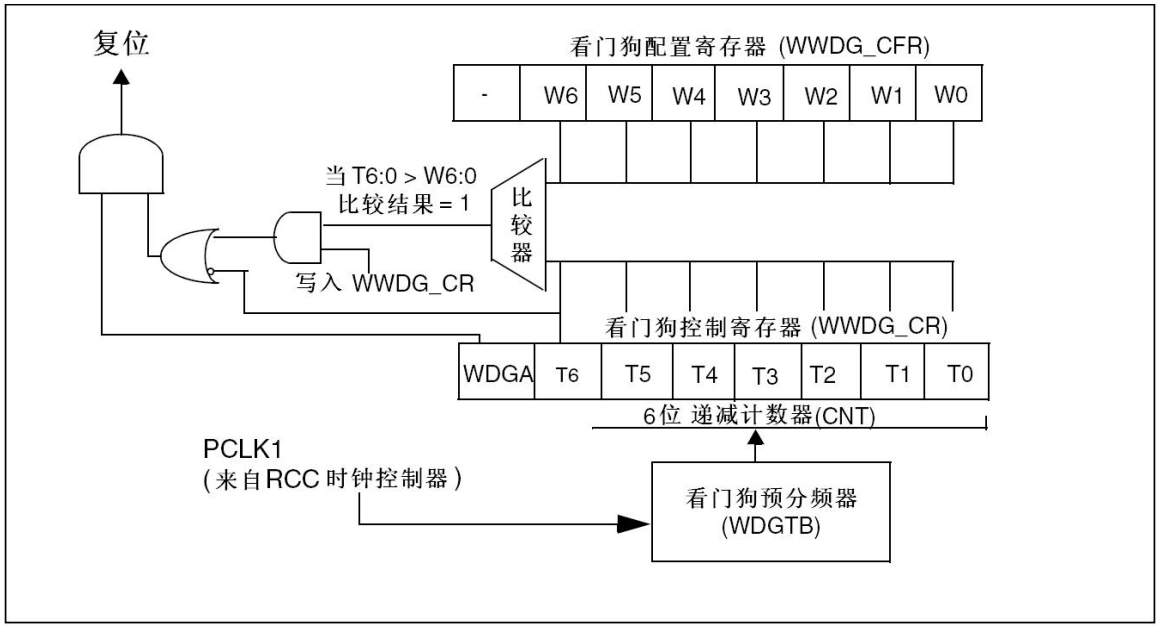
16.2 WWDG 主要特性

- 可编程的自由运行递减计数器
- 条件复位
  - 当递减计数器的值小于 0x40，(若看门狗被启动)则产生复位。
  - 当递减计数器在窗口外被重新装载，(若看门狗被启动)则产生复位。见图 136。
- 如果启动了看门狗并且允许中断，当递减计数器等于 0x40 时产生早期唤醒中断(EWI)，它可以被用于重装载计数器以避免 WWDG 复位。

16.3 WWDG 功能描述

如果看门狗被启动(WWDG_CR 寄存器中的 WDGA 位被置'1')， 并且当 7 位(T[6:0])递减计数器从0x40 翻转到 0x3F(T6 位清零)时，则产生一个复位。如果软件在计数器值大于窗口寄存器中的数值时重新装载计数器，将产生一个复位。

图 135 看门狗框图



应用程序在正常运行过程中必须定期地写入 WWDG_CR 寄存器以防止 MCU 发生复位。只有当计数器值小于窗口寄存器的值时，才能进行写操作。储存在 WWDG_CR 寄存器中的数值必须在0xFF 和 0xC0 之间：

- 启动看门狗 在系统复位后，看门狗总是处于关闭状态，设置 WWDG_CR 寄存器的 WDGA 位能够开启看 门狗，随后它不能再被关闭，除非发生复位。
- 控制递减计数器 递减计数器处于自由运行状态，即使看门狗被禁止，递减计数器仍继续递减

计数。当看门 狗被启用时，T6 位必须被设置，以防止立即产生一个复位。 T[5:0]位包含了看门狗产生复位之前的计时数目；复位前的延时时间在一个最小值和一个最 大值之间变化，这是因为写入 WWDG_CR 寄存器时，预分频值是未知的。 配置寄存器(WWDG_CFR) 中包含窗口的上限值：要避免产生复位，递减计数器必须在其值 小于窗口寄存器的数值并且大于 0x3F 时被重新装载，0 描述了窗口寄存器的工作过程。 另一个重装载计数器的方法是利用早期唤醒中断(EWI)。设置 WWDG_CFR 寄存器中的 WEI 位开启该中断。当递减计数器到达 0x40 时，则产生此中断，相应的中断服务程序(ISR)可以 用来加载计数器以防止 WWDG 复位。在 WWDG_SR 寄存器中写'0'可以清除该中断。

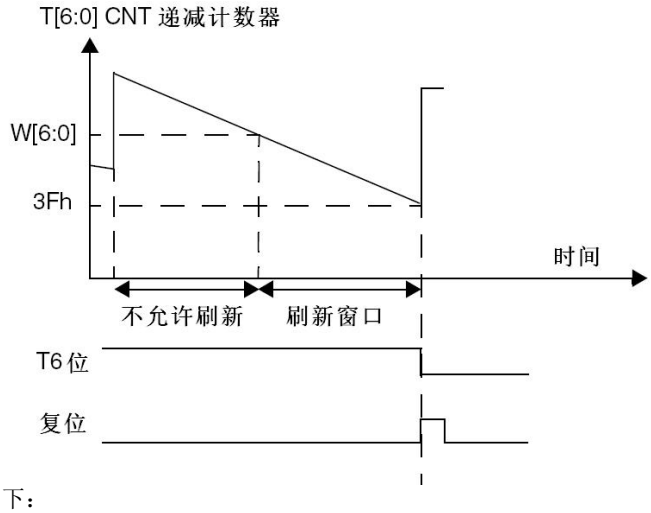
注： 可以用 T6 位产生一个软件复位(设置 WDGA 位为'1'，T6 位为'0')。

16.4 如何编写看门狗超时程序

可以使用 0 提供的公式计算窗口看门狗的超时时间。

警告：当写入WWDG_CR 寄存器时，始终置T6 位为'1'以避免立即产生一个复位。

图 136 窗口看门狗时序图



计算超时的公式如下：

$$T_{WWDG} = T_{PCLK1} \times 4096 \times 2^{WDGTB} \times (T[5:0] + 1); \quad (ms)$$

其中：

T_{WWDG}：WWDG 超时时间 T_{PCLK1}：APB1 以 ms 为单位的时钟间隔

在 PCLK1 = 36MHz 时的最小-最大超时值

WDGTB	最小超时值	最大超时值
0	113μs	7.28ms
1	227μs	14.56ms
2	455μs	29.12ms
3	910μs	58.25ms

16.5 调试模式

当微控制器进入调试模式时(Cortex-M3 核心停止)，根据调试模块中的 DBG_WWDG_STOP 配置位的状态，WWDG 的计数器能够继续工作或停止。详见第 25.15.2 节。

16.6 寄存器描述

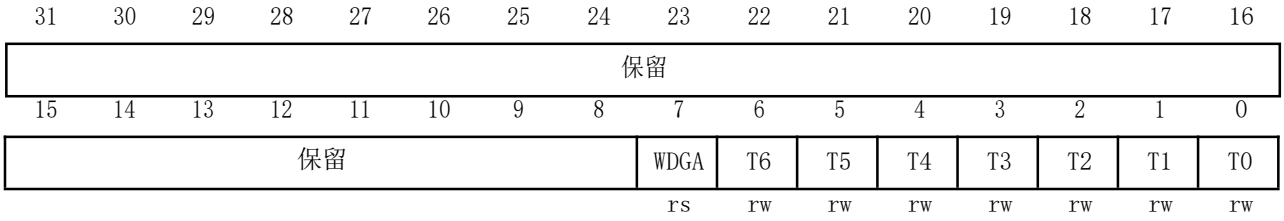
关于在寄存器描述里面所用到的缩写，详见第 2.1 节。

可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

16.6.1 控制寄存器(WWDG_CR)

地址偏移量：0x00

复位值：0x0000 007F

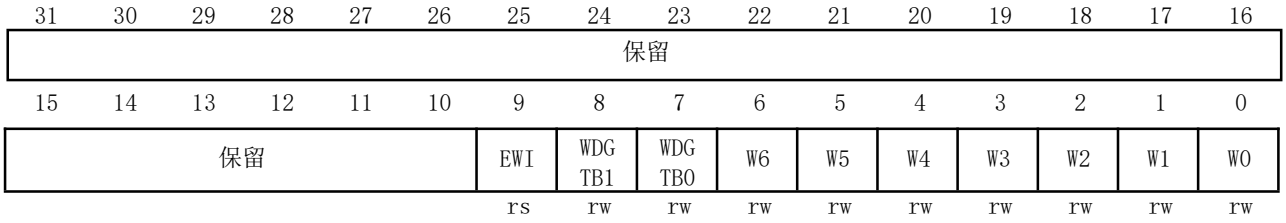


位 31:8	保留。
位 7	**WDGA**: 激活位(Activation bit) 此位由软件置'1'，但仅能由硬件在复位后清'0'。当 WDGA=1 时，看门狗可以产生复位。 0：禁止看门狗 1：启用看门狗
位 6:0	**T[6:0]**: 7 位计数器(MSB 至 LSB) (7-bit counter) 这些位用来存储看门狗的计数器值。每(4096×2^{WDGTB})个 PCLK1 周期减 1。当计数器值从 40h 变为 3Fh 时(T6 变成 0)，产生看门狗复位。

16.6.2 配置寄存器(WWDG_CFR)

地址偏移量：0x04

复位值：0x7F



位 31:8	保留。
位 9	**EWI**: 提前唤醒中断(Early wakeup interrupt) 此位若置'1'，则当计数器值达到 40h，即产生中断。 此中断只能由硬件在复位后清除。
位 8:7	**WDGTB[1:0]**: 时基(Timer base) 预分频器的时基可以设置如下： 00: CK 计时器时钟(PCLK1 除以 4096)除以 1 01: CK 计时器时钟(PCLK1 除以 4096)除以 2 10: CK 计时器时钟(PCLK1 除以 4096)除以 4 11: CK 计时器时钟(PCLK1 除以 4096)除以 8

位 6:0	**W[6:0]**: 7 位窗口值(7-bit window value) 这些位包含了用来与递减计数器进行比较用的窗口值。

16.6.3 状态寄存器(WWDG_SR)

地址偏移量：0x08

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留															EWIF
															rc w0

位 31:1	保留。
位 0	**EWIF**: 提前唤醒中断标志(Early wakeup interrupt flag) 当计数器值达到 40h 时，此位由硬件置'1'。它必须通过软件写'0'来清除。对此位写'1'无效。 若中断未被使能，此位也会被置'1'。

16.6.4 WWDG 寄存器映像

表 62 WWDG 寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	WWDG_CR	保留																								WDGA	T[6:0]						
	复位值																									0	1	1	1	1	1	1	1
004h	WWDG_CFR	保留																						EWI	WDGTB1	WDGTB0	W[6:0]						
	复位值																							0	0	0	1	1	1	1	1	1	1
008h	WWDG_SR	保留																															EWIF
	复位值																																0

有关寄存器的起始地址，参见表 1。

17 USB 全速设备接口（USB）

17.1 USB 简介

USB 外设实现了 USB2.0 全速总线和 APB1 总线间的接口。 USB 外设支持 USB 挂起/恢复操作，可以停止设备时钟实现低功耗。

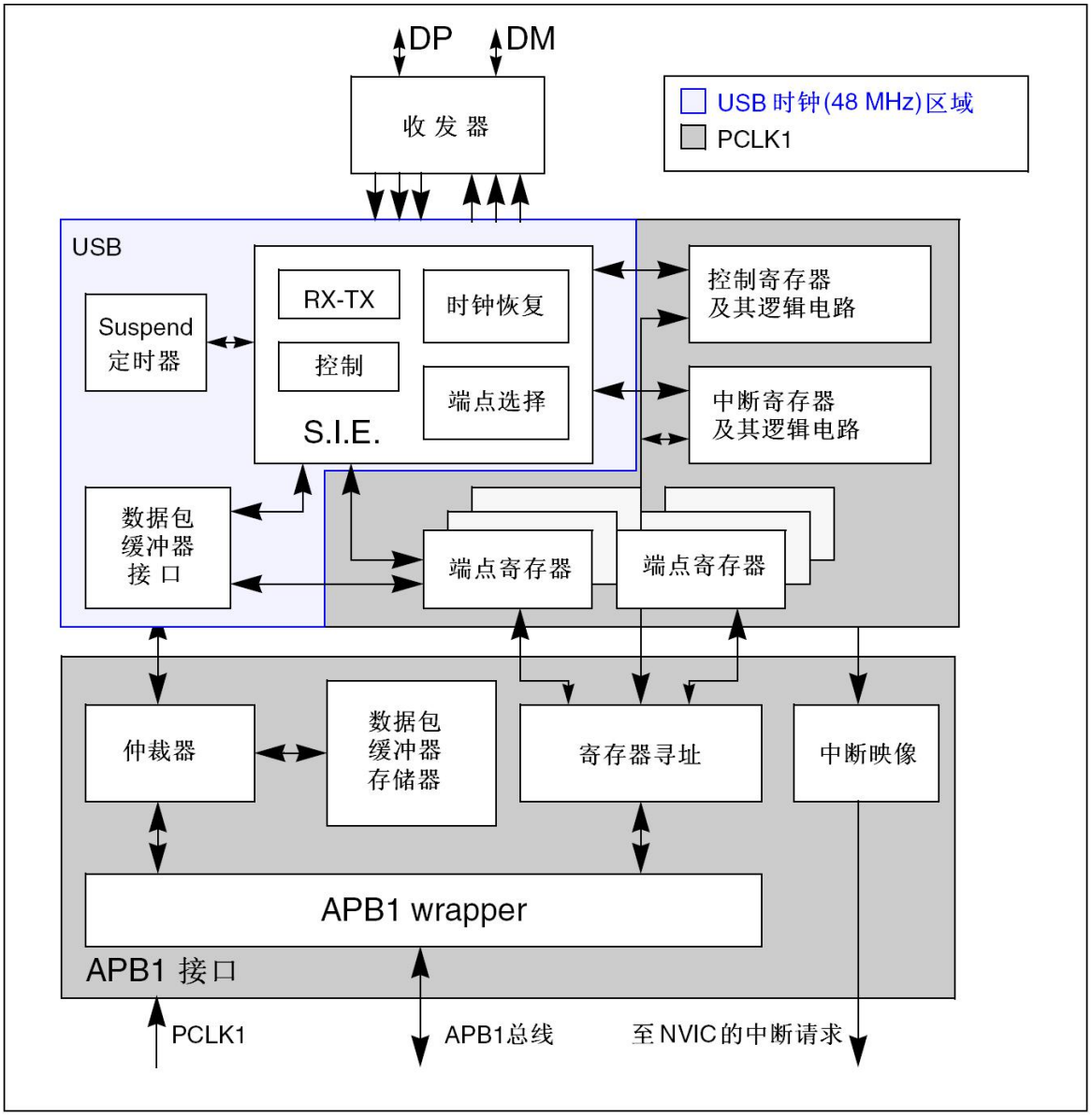
17.2 USB 主要特征

- 符合 USB2.0 全速设备的技术规范
- 可配置 1 到 8 个 USB 端点
- CRC(循环冗余校验)生成/校验，反向不归零(NRZI)编码/解码和位填充
- 支持同步传输
- 支持批量/同步端点的双缓冲区机制
- 支持 USB 挂起/恢复操作
- 帧锁定时钟脉冲生成

注： *USB 和 CAN 共用一个专用的 512 字节的 SRAM 存储器用于数据的发送和接收，因此不能同时使用 USB 和 CAN (共享的 SRAM 被 USB 和 CAN 模块互斥地访问)。USB 和 CAN 可以同时用于一个应用中但不能在同一个时间使用。*

下图是 USB 外设的方框图

图 137 USB 设备框图



17.3 USB 功能描述

USB 模块为 PC 主机和微控制器所实现的功能之间提供了符合 USB 规范的通信连接。PC 主机和微控制器之间的数据传输是通过共享一专用的数据缓冲区来完成的，该数据缓冲区能被 USB 外设直接访问。这块专用数据缓冲区的大小由所使用的端点数目和每个端点最大的数据分组大小所决定，每个端点最大可使用 512 字节缓冲区，最多可用于 16 个单向或 8 个双向端点。USB 模块同 PC 主机通信，根据 USB 规范实现令牌分组的检测，数据发送/接收的处理，和握手分组的处理。整个传输的格式由硬件完成，其中包括 CRC 的生成和校验。

每个端点都有一个缓冲区描述块，描述该端点使用的缓冲区地址、大小和需要传输的字节数。

当 USB 模块识别出一个有效的功能/端点的令牌分组时，(如果需要传输数据并且端点已配置)随之发生相关的数据传输。USB 模块通过一个内部的 16 位寄存器实现端口与专用缓冲区的数据交换。在所有的数据传输完成后，如果需要，则根据传输的方向，发送或接收适当的握手分组。

在数据传输结束时，USB 模块将触发与端点相关的中断，通过读状态寄存器和/或者利用不同的中断处理程序，微控制器可以确定：

- 哪个端点需要得到服务

- 产生如位填充、格式、CRC、协议、缺失 ACK、缓冲区溢出/缓冲区未满等错误时，正在进行的是哪种类型的传输。

USB 模块对同步传输和高吞吐量的批量传输提供了特殊的双缓冲区机制，在微控制器使用一个缓冲区的时候，该机制保证了 USB 外设总是可以使用另一个缓冲区。

在任何不需要使用 USB 模块的时候，通过写控制寄存器总可以使 USB 模块置于低功耗模式(SUSPEND 模式)。在这种模式下，不产生任何静态电流消耗，同时 USB 时钟也会减慢或停止。通过对 USB 线上数据传输的检测，可以在低功耗模式下唤醒 USB 模块。也可以将一特定的中断输入源直接连接到唤醒引脚上，以使系统能立即恢复正常的时钟系统，并支持直接启动或停止时钟系统。

17.3.1 USB 功能模块描述

USB 模块实现了标准 USB 接口的所有特性，它由以下部分组成：

- 串行接口控制器(SIE)：该模块包括的功能有：帧头同步域的识别，位填充，CRC 的产生和校验，PID 的验证/产生，和握手分组处理等。它与 USB 收发器交互，利用分组缓冲接口提供的虚拟缓冲区存储局部数据。它也根据 USB 事件，和类似于传输结束或一个包正确接收等与端点相关事件生成信号，例如帧首(Start of Frame)，USB 复位，数据错误等等，这些信号用来产生中断。
- 定时器：本模块的功能是产生一个与帧开始报文同步的时钟脉冲，并在 3ms 内没有数据传输的状态，检测出(主机的)全局挂起条件。
- 分组缓冲器接口：此模块管理那些用于发送和接收的临时本地内存单元。它根据 SIE 的要求分配合适的缓冲区，并定位到端点寄存器所指向的存储区地址。它在每个字节传输后，自动递增地址，直到数据分组传输结束。它记录传输的字节数并防止缓冲区溢出。
- 端点相关寄存器：每个端点都有一个与之相关的寄存器，用于描述端点类型和当前状态。对于单向和单缓冲器端点，一个寄存器就可以用于实现两个不同的端点。一共 8 个寄存器，可以用于实现最多 16 个单向/单缓冲的端点或者 7 个双缓冲的端点或者这些端点的组合。例如，可以同时实现 4 个双缓冲端点和 8 个单缓冲/单向端点。
- 控制寄存器：这些寄存器包含整个 USB 模块的状态信息，用来触发诸如恢复，低功耗等 USB 事件。
- 中断寄存器：这些寄存器包含中断屏蔽信息和中断事件的记录信息。配置和访问这些寄存器可以获取中断源，中断状态等信息，并能清除待处理中断的状态标志。

注意：　端点 0 总是作为单缓冲模式下的控制端点。

USB 模块通过 APB1 接口部件与 APB1 总线相连，APB1 接口部件包括以下部分：

- 分组缓冲区：数据分组缓存在分组缓冲区中，它由分组缓冲接口控制并创建数据结构。应用软件可以直接访问该缓冲区。它的大小为 512 字节，由 256 个 16 位的字构成。
- 仲裁器：该部件负责处理来自 APB1 总线和 USB 接口的存储器请求。它通过向 APB1 提供较高的访问优先权来解决总线的冲突，并且总是保留一半的存储器带宽供 USB 完成传输。它采用时分复用的策略实现了虚拟的双端口 SRAM，即在 USB 传输的同时，允许应用程序访问存储器。此策略也允许任意长度的多字节 APB1 传输。
- 寄存器映射单元：此部件将 USB 模块的各种字节宽度和位宽度的寄存器映射成能被 APB1 寻址的 16 位宽度的内存集合。
- APB1 封装：此部件为缓冲区和寄存器提供了到 APB1 的接口，并将整个 USB 模块映射到 APB1 地址空间。
- 中断映射单元：将可能产生中断的 USB 事件映射到三个不同的 NVIC 请求线上：
  - USB 低优先级中断(通道 20)：可由所有 USB 事件触发(正确传输，USB 复位等)。固件在处理中断前应当首先确定中断源。
  - USB 高优先级中断(通道 19)：仅能由同步和双缓冲批量传输的正确传输事件触发，目的是保证最大的传输速率。
  - USB 唤醒中断(通道 42)：由 USB 挂起模式的唤醒事件触发。

17.4 编程中需要考虑的问题

在下面的章节中，将介绍 USB 模块和应用程序之间的交互过程，有利于简化应用程序的开发。

17.4.1 通用 USB 设备编程

这一部分描述了实现 USB 设备功能的应用程序需要完成的任务。除了介绍一般的 USB 事件中应该采取的操作外，还着重介绍了双缓冲端点和同步传输的操作。这些相关的操作都是由 USB 模块初始化，并由以下几节所描述的 USB 事件所驱动。

17.4.2 系统复位和上电复位

发生系统复位或者上电复位时，应用程序首先需要做的是提供 USB 模块所需要的时钟信号，然后清除复位信号，使程序可以访问 USB 模块的寄存器。复位之后的初始化流程如下所述：

首先，由应用程序激活寄存器单元的时钟，再配置设备时钟管理逻辑单元的相关控制位，清除复位信号。

其次，必须配置 CNTR 寄存器的 PDWN 位用以开启 USB 收发器相关的模拟部分，这点需要特别的处理。此位能打开为端点收发器供电的内部参照电压。由于打开内部电压需要一段启动时间(数据手册中的 $t_{STARTUP}$)，在此期间内 USB 收发器处于不确定状态，所以在设置 CNTR 寄存器的PDWN 后必需等待一段时间之后，才能清除 USB 模块的复位信号(清除 CNTR 寄存器上的 FRES位)，和 ISTR 寄存器的内容，以便在使能其他任何单元的操作之前清除未处理的假中断标志。

最后，应用程序需要通过配置设备时钟管理逻辑的相应控制位来为 USB 模块提供标准所定义的 48MHz 时钟。

当系统复位时，应用程序应该初始化所有需要的寄存器和分组缓冲区描述表，使 USB 模块能够产生正常的中断和完成数据传输。所有与端点无关的寄存器需要根据应用的需求进行初始化(比如中断使能的选择，分组缓冲区地址的选择等)。接下来按照 USB 复位处理(参见下段)。

USB 复位(RESET 中断)

发生 USB 复位时，USB 模块进入前面章节中描述过的系统复位状态：所有端点的通信都被禁止(USB 模块不会响应任何分组)。在 USB 复位后，USB 模块被使能，同时地址为 0 的默认控制端点 (端点 0)也需要被使能。这可以通过配置 USB_DADDR 寄存器的 EF 位，EP0R 寄存器和相关的分组缓冲区来实现。在 USB 设备的枚举阶段，主机将分配给设备一个唯一的地址，这个地址必须写入 USB_DADDR 寄存器的 ADD[6:0]位中，同时配置其他所需的端点。

当复位中断产生时，应用程序必需在中断产生后的 10ms 之内使能端点 0 的传输。

分组缓冲区的结构和用途

每个双向端点都可以接收或发送数据。接收到的数据存储在该端点指定的专用缓冲区内，而另一个缓冲区则用于存放待发送的数据。对这些缓冲区的访问由分组缓冲区接口模块实现，它提出缓冲区访问请求，并等待确认信息后返回。为防止产生微控制器与 USB 模块对缓冲区的访问冲突，缓冲区接口模块使用仲裁机制，使 APB1 总线的一半周期用于微控制器的访问，另一半保证 USB 模块的访问。这样，微控制器和 USB 模块对分组缓冲区的访问如同对一个双端口 SRAM 的访问，即使微控制器连续访问缓冲区，也不会产生访问冲突。

USB 模块使用固定的时钟，按照 USB 标准，此时钟频率被固定为 48MHz。APB1 总线的时钟可以 大于或者小于这个频率。

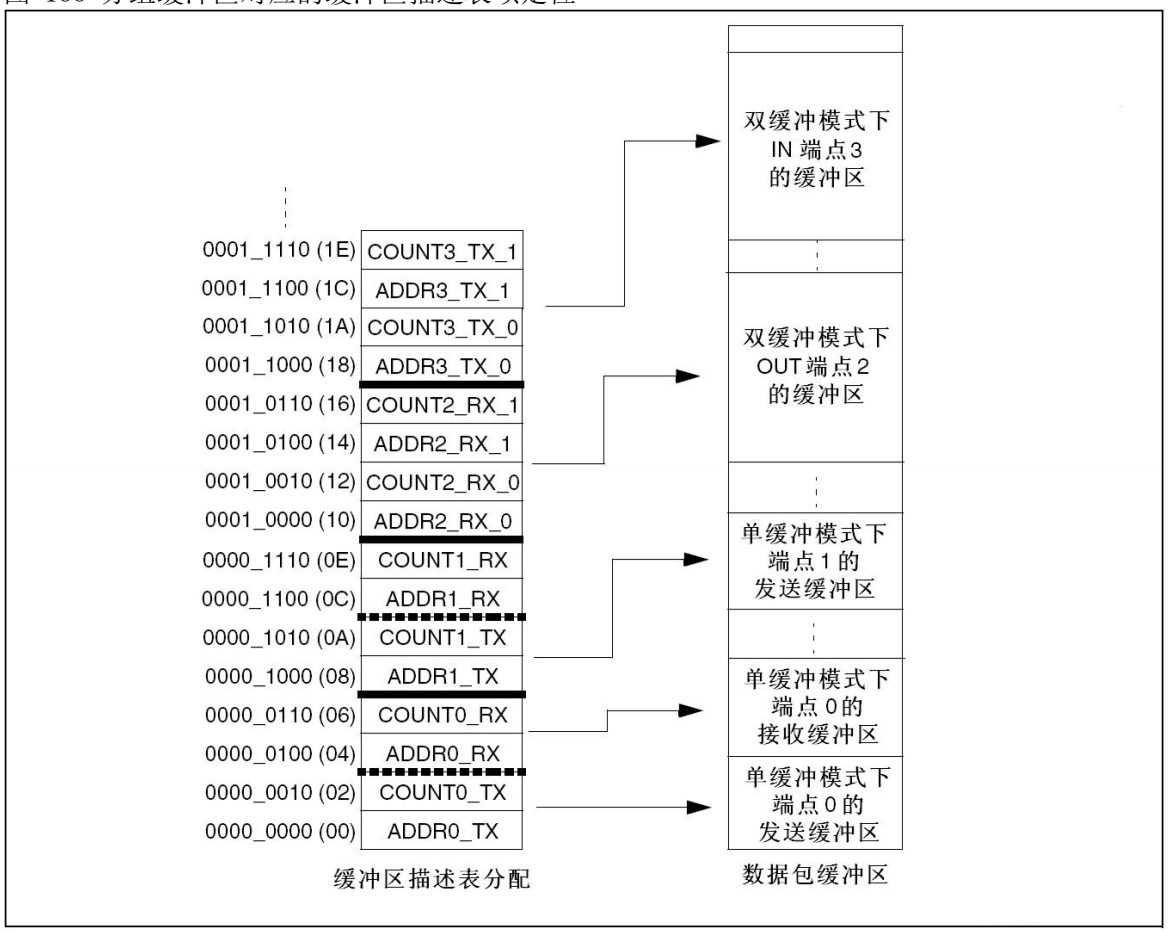
注意：为满足 USB 数据传输率和分组缓冲区接口的系统需求，APB1 总线时钟的频率必须大于 8MHz， 以避免数据缓冲区溢出或不满

每个端点对应于两个分组缓冲区(一般一个用于发送，另一个用于接收)。这些缓冲区可以位于整个分组存储区的任意位置，因为它们的地址和长度都定义在缓冲区描述表中，而缓冲区描述表也同样位于分组缓冲区中，其地址由 USB_BTABLE 寄存器确定。

缓冲区描述表的每个表项都关联到一个端点寄存器，它由 4 个 16 位的字组成，因此缓冲区描述

表的起始地址按 8 字节对齐(寄存器的最低 3 位总是'000')。第 18.5.3 节详细介绍缓冲区描述表表项。如果是非同步非双缓冲的单向端点，只需要一个分组缓冲区(即发送方向上的分组缓冲区)。其他未用到的端点或某个未使用的方向上的缓冲区描述表项可以用于其他用途。同步和双缓冲批量端点有特殊的分组缓冲区处理方法(请分别参考第 18.4.4 节：同步传输和第 18.4.3 节：双缓冲端点)。下图描述了缓冲区描述表项和分组缓冲区区域的关系。

图 138 分组缓冲区对应的缓冲区描述表项定位



不管是接收还是发送，分组缓冲区都是从底部开始使用的。USB 模块不会改变超出当前分配到的缓冲区区域以外的其他缓冲区的内容。如果缓冲区收到一个比自己大的数据分组，它只会接收最大为自身大小的数据，其他的丢掉，即发生了所谓的缓冲区溢出异常。

端点初始化

初始化端点的第一步是把适当的值写到 ADDRn_TX 或 ADDRn_RX 寄存器中，以便 USB 模块能找到要传输的数据或准备好接收数据的缓冲区。USB_EPnR 寄存器的 EP_TYPE 位确定端点的基本类型，EP_KIND 位确定端点的特殊特性。作为发送方，需要设置 USB_EPnR 寄存器的 STAT_TX 位来使能端点，并配置 COUNTn_TX 位决定发送长度。作为接收方，需要设置 STAT_RX 位来使能端点，并且设置 BL_SIZE 和 NUM_BLOCK 位，确定接收缓冲区的大小，以检测缓冲区溢出的异常。对于非同步非双缓冲批量传输的单向端点，只需要设置一个传输方向上的寄存器。一旦端点被使能，应用程序就不能再修改 USB_EPnR 寄存器的值和 ADDRn_TX / ADDRn_RX, COUNTn_TX / COUNTn_RX 所在的位置，因为这些值会被硬件实时修改。当数据传输完成时，CTR 中断会产生，此时上述寄存器可以被访问，并重新使能新的传输。

IN 分组(用于数据发送)

当接收到一 IN 令牌分组时，如果接收到的地址和一个配置好的端点地址相符合的话，USB 模块将会根据缓冲区描述表的表项，访问相应的 ADDRn_TX 和 COUNTn_TX 寄存器，并将这些

寄存器中的数值存储到内部的 16 位寄存器 ADDR 和 COUNT(应用程序无法访问)中。此时，USB 模块开始根据 DTOG_TX 位发送 DATA0 或 DATA1 分组，并访问缓冲区(请参考'分组缓冲区的结构和用途'段落)。在 IN 分组传输完毕之后，从缓冲区读到的第一个字节将被装载到输出移位寄存器中，并开始发送。最后一个数据字节发送完成之后，计算好的 CRC 将被发送。如果收到的分组所对应的端点是无效的，将根据 USB_EPnR 寄存器上的 STAT_TX 位发送 NAK 或 STALL 握手分组而不发送数据。

ADDR 内部寄存器被用作当前缓冲区的指针，COUNT 寄存器用于记录剩下未传输的字节数。USB 总线使用低字节在先的方式传输从缓冲区中读出的数据。数据从 ADDRn_TX 指向的数据分组缓冲区开始读取，长度为 COUNTn_TX/2 个字。如果发送的数据分组为奇数个字节，则只使用最后一个字的低 8 位。

在接收到主机响应的 ACK 后，USB_EPnR 寄存器的值有以下更新：DTOG_TX 位被翻转，STAT_TX 位为'10'，使端点无效，CTR_TX 位被置位。应用程序需要通过 USB_ISTR 寄存器的 EP_ID 和 DIR 位识别产生中断的 USB 端点。CTR_TX 事件的中断服务程序需要首先清除中断标志位，然后准备好需要发送的数据缓冲区，更新 COUNTn_TX 为下次需要传输的字节数，最后再设置 STAT_TX 位为'11'(端点有效)，再次使能数据传输。当 STAT_TX 位为'10'时(端点为 NAK 状态)，任何发送到该端点的 IN 请求都会被 NAK，USB 主机会重发 IN 请求直到该端点确认请求有效。上述操作过程是必需遵守的，以避免丢失紧随上一次 CTR 中断请求的下一个 IN 传输请求。

OUT 分组和 SETUP 分组(用于数据接收)

USB 模块对这两种分组的处理方式基本相同；对 SETUP 分组的特殊处理将在下面关于控制传输部分详细说明。当接收到一个 OUT 或 SETUP 分组时，如果地址和某个有效端点的地址相匹配，USB 模块将访问缓冲区描述表，找到与该端点相关的 ADDRn_RX 和 COUNTn_RX 寄存器，并将 ADDRn_RX 寄存器的值保存在内部 ADDR 寄存器中。同时，COUNT 会被复位，从 COUNTn_RX 中读出的 BL_SIZE 和 NUM_BLOCK 的值用于初始化内部 16 位寄存器 BUF_COUNT，该寄存器用于检测缓冲区溢出(所有的内部寄存器都不能被应用程序访问)。USB 模块将随后收到的数据按字方式组织(先收到的为低字节)，并存储到 ADDR 指向的分组缓冲区中。同时，BUF_COUNT 值自动递减，COUNT 值自动递增。当检测到数据分组的结束信号时，USB 模块校验收到 CRC 的正确性。如果传输中没有任何错误发生，则发送 ACK 握手分组到主机。即使发生 CRC 错误或者其他类型的错误(位填充，帧错误等)，数据还是会被保存到分组缓冲区中，至少会保存到发生错误的数据点，只是不会发送 ACK 分组，并且 USB_ISTR 寄存器的 ERR 位将会置位。在这种情况下，应用程序通常不需要干涉处理，USB 模块将从传输错误中自动恢复，并为下一次传输做好准备。如果收到的分组所对应的端点没有准备好，USB 模块将根据 USB_EPnR 寄存器的 STAT_RX 位发送 NAK 或 STALL 分组，数据将不会被写入接收缓冲区。

ADDRn_RX 的值决定接收缓冲区的起始地址，长度由包含 CRC 的数据分组的长度(即有效数据长度+2)决定，但不能超过 BL_SIZE 和 NUM_BLOCK 所定义的缓冲区的长度。如果接收到的数据分组的长度超出了缓冲区的范围，超过范围的数据不会被写入缓冲区，USB 模块将报告缓冲区发生溢出，并向主机发送 STALL 握手分组，通知此次传输失败，也不产生中断。

如果传输正确完成，USB 模块将发送 ACK 握手分组，内部的 COUNT 寄存器的值会被复制到相应的 COUNTn_RX 寄存器中，BL_SIZE 和 NUM_BLOCK 的值保持不变，也不需要重写。USB_EPnR 寄存器按下列方式更新：DTOG_RX 位翻转，STAT_RX=10(NAK)使端点无效，CTR_RX 位置位(如果 CTR 中断已使能，将触发中断)。如果传输过程中发生了错误或者缓冲区溢出，前面所列出的动作都不会发生。CRT 中断发生时，应用程序需要首先根据 USB_ISTR 寄存器的 EP_ID 和 DIR 位识别是哪个端点的中断请求。在处理 CTR_RX 中断事件时，应用程序首先要确定传输的类型(根据 USB_EPnR 寄存器的 SETUP 位)，同时清除中断标志位，然后读相关的缓冲区描述表表项指向的 COUNTn_RX 寄存器，获得此次传输的总字节数。处理完接收到的数据后，应用程序需要将 USB_EPnR 中的 STAT_RX 位置成'11'，使能下一次的的传输。当 STAT_RX 位为'10'时(NAK)，任何一个发送到端点上的 OUT 请求都会被 NAK，PC 主机将不断重发被 NAK 的分组，直到收到端点的 ACK 握手分组。以上描述的操作次序是必需遵守的，以避免丢失紧随上一个 CTR 中断的另一个 OUT 分组请求。

控制传输

控制传输由 3 个阶段组成，首先是主机发送 SETUP 分组的 SETUP 阶段，然后是主机发送零个或多个数据的数据阶段，最后是状态阶段，由与数据阶段方向相反的数据分组构成。SETUP 传输只发生在控制端点，它非常类似于 OUT 分组的传输过程。使能 SETUP 传输除了需要分别初始化 DTOG_TX 位为'1'，DTOG_RX 位为'0'外，还需要设置 STAT_TX 位和 STAT_RX 位为 10(NAK)， 由应用程序根据 SETUP 分组的相应字段决定后面的传输是 IN 还是 OUT。控制端点在每次发生 CTR_RX 中断时，都必须检查 USB_EPnR 寄存器的 SETUP 位，以识别是普通的 OUT 分组还是SETUP 分组。USB 设备应该能够通过 SETUP 分组中的相应数据决定数据阶段传输的字节数和方向，并且能在发生错误的情况下发送 STALL 分组，拒绝数据的传输。因此在数据阶段，未被使用到的方向都应该被设置成 STALL，并且在开始传输数据阶段的最后一个数据分组时，其反方向的传输仍设成 NAK 状态，这样，即使主机立刻改变了传输方向(进入状态阶段)，仍然可以保持为等待控制传输结束的状态。在控制传输成功结束后，应用程序可以把 NAK 变为 VALD，如果控制传输出错，就改为 STALL。此时，如果状态分组是由主机发送给设备的，那么 STATUS_OUT 位(USB_EPnR 寄存器中的 EP_KIND)应该被置位，只有这样，在状态传输过程中收到了非零长度的数据分组，才会产生传输错误。在完成状态传输阶段后，应用程序应该清除 STATUS_OUT 位，并且将 STAT_RX 设为 VALID 表示已准备好接收一个新的命令请求，STAT_TX 则设为 NAK， 表示在下一个 SETUP 分组传输完成前，不接受数据传输的请求。

USB 规范定义 SETUP 分组不能以非 ACK 握手分组来响应，如果 SETUP 分组传输失败，则会引发下一个 SETUP 分组。因此，以 NAK 或 STALL 分组响应主机的 SETUP 分组是被禁止的。

当 STAT_RX 位被设置为'01'(STALL)或'10'(NAK)时，如果收到 SETUP 分组，USB 模块会接收分组，开始分组所要求的数据传输，并回送 ACK 握手分组。如果应用程序在处理前一个 CTR_RX 事件时 USB 模块又收到了 SETUP 分组(即 CTR_RX 仍然保持置位)，USB 模块会丢掉收到的 SETUP 分组，并且不回答任何握手分组，以此来模拟一个接收错误，迫使主机再次发送 SETUP 分组。这样做是为了避免丢失紧随一次 CTR_RX 中断之后的又一个 SETUP 分组传输。

17.4.3 双缓冲端点

USB 标准不仅为不同的传输模式定义了不同的端点类型，而且对这些数据传输所需要的系统要求做了描述。其中，批量端点适用于在主机 PC 和 USB 设备之间传输大批量的数据，因为主机可以在一帧内利用尽可能多的带宽批量传输数据，使传输效率得到提高。然而，当 USB 设备处理前一次的数据传输时，又收到新的数据分组，它将回应 NAK 分组，使 PC 主机不断重发同样的数据分组，直到设备在可以处理数据时回应 ACK 分组。这样的重传占用了很多带宽，影响了批量传输的速率，因此引入了批量端点的双缓冲机制，提高数据传输率。

使用双缓冲机制时，单向端点的数据传输将使用到该端点的接收和发送两块数据缓冲区。数据翻转位用来选择当前使用到两块缓冲区中的哪一块，使应用程序可以在 USB 模块访问其中一块缓冲区的同时，对另一块缓冲区进行操作。例如，对一个双缓冲批量端点进行 OUT 分组传输时，USB 模块将来自 PC 主机的数据保存到一个缓冲区，同时应用程序可以对另一个缓冲区中的数据进行处理(对于 IN 分组来说，情况是一样的)。

因为切换缓冲区的管理机制需要用到所有 4 个缓冲区描述表的表项，分别用来表示每个方向上的两个缓冲区的地址指针和缓冲区大小，因此用来实现双缓冲批量端点的 USB_EPnR 寄存器必需配置为单向。所以只需要设定 STAT_RX 位(作为双缓冲批量接收端点)或者 STAT_TX 位(作为双缓冲批量发送端点)。如果需要一个双向的双缓冲批量端点，则须使用两个 USB_EPnR 寄存器。

为尽可能利用双缓冲的优势，达到较高的传输速率，双缓冲批量端点的流量控制流程与其他端点的稍有不同。它只在缓冲区发生访问冲突时才会设置端点为 NAK 状态，而不是在每次传输成功后都将端点设为 NAK 状态。

DTOG 位用来标识 USB 模块当前所使用的储存缓冲区。双缓冲批量端点接收方向的缓冲区由 DTOG_RX(USB_EPnR 寄存器的第 14 位)标识，而双缓冲批量端点发送方向的缓冲区由 DTOG_TX(USB_EPnR 寄存器的第 6 位)标识。同时，USB 模块也需要知道当前哪个缓冲区正在被应用程序使用，以避免发生冲突。由于 USB_EPnR 寄存器中有 2 个 DTOG 位，而 USB 模块只使用其中的一位来标识硬件所使用的缓冲区，因此，应用程序可使用另一位来

标识当前正在使用哪个缓冲区，这个新的标识被称为 SW_BUF 位。下表列出了双缓冲批量端点在实现发送和接收操作时，USB_EPNR 寄存器的 DTOG 位和 SW_BUF 位之间的关系。

表 63 双缓冲批量端点缓冲区标识定义

缓冲区标识位	作为发送端点	作为接收端点
DTOG	DTOG_TX(USB_EPnR 寄存器的第 6 位)	DTOG_RX(USB_EPnR 寄存器的第 14 位)
SW_BUF	USB_EPnR 寄存器的第 14 位	USB_EPnR 寄存器的第 6 位

USB 模块当前使用的缓冲区由 DTOG 位标识，而应用程序所使用的缓冲区由 SW_BUF 位标识，这两个位的标识方式相同，下表描述了这种标识方式。

表 64 双缓冲批量端点的缓冲区使用标识

端点类型	DTOG 位	SW_BUF 位	USB 模块使用的缓冲区	应用程序使用的缓冲区
IN 端点	0	1	ADDRn_TX_0 / COUNTn_TX_0	ADDRn_TX_1 / COUNTn_TX_1
	1	0	ADDRn_TX_1 / COUNTn_TX_1	ADDRn_TX_0 / COUNTn_TX_0
	0	0	无(1)	ADDRn_TX_0 / COUNTn_TX_0
	1	1	无(1)	ADDRn_TX_0 / COUNTn_TX_0
OUT 端点	0	1	ADDRn_RX_0 / COUNTn_RX_0	ADDRn_RX_1 / COUNTn_RX_1
	1	0	ADDRn_RX_1 / COUNTn_RX_1	ADDRn_RX_0 / COUNTn_RX_0
	0	0	无(1)	ADDRn_RX_0 / COUNTn_RX_0
	1	1	无(1)	ADDRn_RX_0 / COUNTn_RX_0

1. 端点处于 NAK 状态

可以通过以下方式设置一个双缓冲批量端点：

- 将 USB_EPnR 寄存器的 EP_TYPE 位设为’00’，定义端点为批量端点
- 将 USB_EPnR 寄存器的 EP_KIND 位设为’1’，定义端点为双缓冲端点

应用程序根据传输开始时用到的缓冲区来初始化 DTOG 和 SW_BUF 位；这需要考虑到这两位的数据翻转特性。设置好 DBL_BUF 位之后，每完成一次传输后，USB 模块将根据双缓冲批量端点的流量控制操作，并且持续到 DBL_BUF 变为无效为止。每次传输结束，根据端点的传输方向，CTR_RX 位或 CTR_TX 位将会置为’1’。与此同时，硬件将设置相应的 DTOG 位，完全独立于软件来实现缓冲区交换机制。DBL_BUF 位设置后，每次传输结束时，双缓冲批量端点的 STAT 位的取值不会像其他类型端点一样受到传输过程的影响，而是一直保持为’11’(有效)。但是，如果在收到新的数据分组的传输请求时，USB 模块和应用程序发生了缓冲区访问冲突(即 DTOG 和 SW_BUF 为相同的值，见表 64)，状态位将会被置为’10’(NAK)。应用程序响应 CTR 中断时，首先要清除中断标志，然后再处理传输完成的数据。应用程序访问缓冲区之后，需要翻转 SW_BUF 位，以通知 USB 模块该块缓冲区已变为可用状态。由此，双缓冲批量传输的 NAK 分组的数目只由应用程序处理一次数据传输的快慢所决定：如果数据处理的时间小于 USB 总线上完成一次数据传输的时间，则不会发生重传，此时，数据的传输率仅受限于 USB 主机。

应用程序也可以不考虑双缓冲批量端点的特殊控制流程，直接在相应 USB_EPnR 寄存器的 STAT 位写入非’11’的任何状态，在这种情况下，USB 模块将按照写入的状态执行流程而忽略缓冲器实际的使用情况。

17.4.4 同步传输

USB 标准定义了一种全速的需要保持固定和精确的数据传输率的传输方式：同步传输。同步传输一般用于传输音频流、压缩的视频流等对数据传输率有严格要求的数据。一个端点如果在枚举时被定义为“同步端点”，USB 主机则会为每个帧分配固定的带宽，并且保证每个帧正好传送一个 IN 分组或者 OUT 分组(由端点传输方向确定分组类型)。为了满足带宽要求，同步传输中没有出错重传；这也就意味着，同步传输在发送或接收数据分组之后，无握手协议，即不会发送ACK 分组。同样，同步传输只传送 PID(分组 ID)为 DATA0 的数据包，而不会用到数据翻

转机制。

通过设置 USB_EPnR 寄存器 EP_TYPE 为’10’，可以使其成为同步端点。同步端点没有握手机制，根据 USB 标准中的说明，USB_EPnR 寄存器的 STAT_RX 位和 STAT_TX 位分别只能设成’00’(禁止)和’11’(有效)。同步传输通过实现双缓冲机制来简化软件应用程序开发，它同样使用两个缓冲区，以确保在 USB 模块使用其中一块缓冲区时，应用程序可以访问另外一块缓冲区。

USB 模块使用的缓冲区根据不同的传输方向，由不同的 DTOG 位来标识。(同一寄存器中的 DTOG_RX 位用来标识接收同步端点，DTOG_TX 位用来标识发送同步端点)，见下表。

表 65 同步端点的缓冲区使用标识

端点类型	DTOG 位值	USB 模块使用的缓冲区	应用程序使用的缓冲区
IN 端点	0	ADDRn_TX_0 / COUNTn_TX_0	ADDRn_TX_1 / COUNTn_TX_1
	1	ADDRn_TX_1 / COUNTn_TX_1	ADDRn_TX_0 / COUNTn_TX_0
OUT 端点	0	ADDRn_RX_0 / COUNTn_RX_0	ADDRn_RX_1 / COUNTn_RX_1
	1	ADDRn_RX_1 / COUNTn_RX_1	ADDRn_RX_0 / COUNTn_RX_0

与双缓冲批量端点一样，一个 USB_EPnR 寄存器只能处理同步端点单方向的数据传输，如果要求同步端点在两个传输方向上都有效，则需要使用两个 USB_EPnR 寄存器。

应用程序需要根据首次传输的数据分组来初始化 DTOG 位；它的取值还需要考虑到 DTOG_RX 或 DTOG_TX 两位的数据翻转特性。每次传输完成时，USB_EPnR 寄存器的 CTR_RX 位或 CTR_TX 位置位。与此同时，相关的 DTOG 位由硬件翻转，从而使得交换缓冲区的操作完全独立于应用程序。传输结束时，STAT_RX 或 STAT_TX 位不会发生变化，因为同步传输没有握手机制，所以不需要任何流量控制，而一直设为’11’(有效)。同步传输中，即使 OUT 分组发生 CRC 错误或者缓冲区溢出，本次传输仍被看作是正确的，并且可以触发 CTR_RX 中断事件；但是，发生 CRC 错误时硬件会设置 USB_ISTR 寄存器的 ERR 位，提醒应用程序数据可能损坏。

17.4.5 挂起/恢复事件

USB 标准中定义了一种特殊的设备状态，即挂起状态，在这种状态下 USB 总线上的平均电流消耗不超过 500uA。这种电流限制对于由总线供电的 USB 设备至关重要，而自供电的设备则不需要严格遵守这样的电流消耗限制。USB 主机以 3 毫秒内不发送任何信号标志进入挂起状态。通常情况下 USB 主机每毫秒会发送一个 SOF，当 USB 模块检测到 3 个连续的 SOF 分组丢失事件即可判定主机发出了挂起请求，接着它会置位 SB_ISTR 寄存器的 SUSP 位，以触发挂起中断。USB 设备进入挂起状态之后，将由“唤醒”序列唤醒。所谓的“唤醒”序列， 可以由 USB 主机发起，也 可以由 USB 设备本身触发；但是，只有 USB 主机可以结束“唤醒”序列。被挂起的 USB 模块必须至少还具备检测 RESET 信号的功能，它会将其当作一次正常的复位操作来执行。

实际的挂起操作过程对于不同的 USB 设备来说是不同的，因为需要不同的操作来降低电源消耗。下面描述了一起典型的挂起操作，重点介绍应用程序如何响应 USB 模块的 SUSP 信号。

1. 将 USB_CNTR 寄存器的 FSUSP 置为’1’，这将使 USB 模块进入挂起状态。USB 模块一旦进入挂起状态，对 SOF 的检测立刻停止，以避免在 USB 挂起时又发生新的 SUSP 事件。
2. 消除或减少 USB 模块以外的其他模块的静态电流消耗。
3. 将 USB_CNTR 寄存器的 LP_MODE 位置为’1’，这将消除模拟 USB 收发器的静态电流消耗，但仍能检测到唤醒信号。
4. 可以选择关闭外部振荡器和设备的 PLL，以停止设备内部的任何活动。

当设备处于挂起状态时发生 USB 事件，该设备会被唤醒，并需要调用“唤醒”例程来恢复系统时钟，和 USB 数据传输。如果唤醒设备的是 USB 复位操作，则应该保证唤醒的过程不要超过 10 毫秒(参见“USB 协议规范”)。USB 模块处于挂起状态时，唤醒或复位事件需要清除 USB_CNTR 寄存器的 LP_MODE 位。即使唤醒事件可以立刻触发一个 WKUP 中断事件，但由于恢复系统时钟需要比较长的延迟时间，处理 WKUP 中断的中断服务程序必须非常小心；

为了减短系统唤醒的时间，建议将唤醒代码直接写在挂起代码后面，这样就可以在系统时钟重启后迅速进入唤醒代码中执行。为防止或减少 ESD 等干扰意外地唤醒系统(从挂起模式退出是一个异步 事件)，在挂起过程中数据线被过滤，滤波宽度大约为 70nS。

下面是唤醒操作的过程：启动外部振荡器和设备的 PLL(此项可选)。

1. 清零 USB_CNTR 寄存器的 FSUSP 位。
2. USB_FNR 寄存器的 RXDP 和 RXDM 位可以用来判断是什么触发了唤醒事件，如表 66 所示，它还同时列出了各种情况软件应该采取的操作。如果需要的话，可以通过检测这两 位变成'10'(代表空闲总线状态)的时间来知道唤醒或复位事件的结束。此外，在复位事件结束时，USB_ISTR 寄存器的 RESET 位被置为'1'，如果 RESET 中断被使能，就会产生中断。此中断应该按正常的复位操作处理。

表 66 唤醒事件检测

[RXDP，RXDM]的状态	唤醒事件	应用程序应执行的操作
00	复位	无
10	无(总线干扰)	恢复到挂起状态
01	恢复挂起	无
11	未定义的值(总线干扰)	恢复到挂起状态

设备可能不是被与 USB 模块相关的事件唤醒的(例如一个鼠标的移动可唤醒整个系统)。在这种情况下，先将 USB_CNTR 寄存器的 RESUME 位置为'1'，然后在 1ms－15ms 之间再把它清为 0 可以启动唤醒序列(这个间隔可以用 ESOF 中断来实现，该中断在内核正常运行时每 1ms 发生一次)。RESUME 位被清零后，唤醒过程将由主机 PC 完成，可以利用 USB_FNR 寄存器的RXDP 和RXDM 位来判断唤醒是否完成。

注意： *只有在 USB 模块被设置为挂起状态时(设置 USB_CNTR 寄存器的 FSUSP 位为'1')，才可以设置 RESUME 位。*

17.5 USB 寄存器描述

USB 模块的寄存器有以下三类:：

- 通用类寄存器：中断寄存器和控制寄存器
- 端点类寄存器：端点配置寄存器和状态寄存器
- 缓冲区描述表类寄存器：用来确定数据分组存放地址的寄存器 缓冲区描述表类寄存器的基地址由 USB_BTABLE 寄存器指定，所有其他寄存器的基地址则为

USB 模块的基地址 0x4000 5C00。由于 APB1 总线按 32 位寻址，因此所有的 16 位寄存器的地址都是按 32 位字对齐的。同样的地址对齐方式也用于从 0x4000 6000 开始的分组缓冲存储区。
关于寄存器描述中使用的一些缩略语，请参考第 2.1 节。 可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

17.5.1 通用寄存器

这组寄存器用于定义 USB 模块的工作模式，中断的处理，设备的地址和读取当前帧的编号。

USB 控制寄存器(USB_CNTR)

地址偏移：0x40

复位值：0x0003

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRM	PMA OVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	保留			RESUME	FSUSP	LP MODE	PDMN	FRES
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

位 15	**CTRM**：正确传输(CTR)中断屏蔽位(Correct transfer interrupt mask) 0：正确传输(CTR)中断禁止 1：正确传输(CTR)中断使能，在中断寄存器的相应位被置 1 时产生中断。

位 14	**PMAOVRM**：分组缓冲区溢出中断屏蔽位(Packet memory area over / underrun interrupt mask) 0：PMAOVR 中断禁止 1：PMAOVR 中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 13	**ERRM**：出错中断屏蔽位(Error interrupt mask) 0：出错中断禁止 1：出错中断使能，在中断寄存器的相应位被置 1 时产生中断。

位 12	**WKUPM**：唤醒中断屏蔽位(Wakeup interrupt mask) 0：唤醒中断禁止 1：唤醒中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 11	**SUSPM**：挂起中断屏蔽位(Suspend mode interrupt mask) 0：挂起(SUSP)中断禁止 1：挂起(SUSP)中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 10	**RESETM**：USB 复位中断屏蔽位(USB reset interrupt mask) 0：USB RESET 中断禁止 1：USB RESET 中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 9	**SOFM**：帧首中断屏蔽位(Start of frame interrupt mask) 0：SOF 中断禁止 1：SOF 中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 8	**ESOFM**：期望帧首中断屏蔽位(Expected start of frame interrupt mask) 0：ESOF 中断禁止 1：ESOF 中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 4	**RESUME**：唤醒请求(Resume request) 设置此位将向 PC 主机发送唤醒请求。根据 USB 协议，如果此位在 1ms 到 15ms 内保持有效，主机将对 USB 模块实行唤醒操作。
位 3	**FSUSP**：强制挂起(Force suspend) 当 USB 总线上保持 3ms 没有数据通信时，SUSP 中断会被触发，此时软件必需设置此位。 0：无效 1：进入挂起模式，USB 模拟收发器的时钟和静态功耗仍然保持。如果需要进入低功耗状态(总线供电类的设备)，应用程序需要先置位 FSUSP 再置位 LP_MODE。
位 2	**LP_MODE**：低功耗模式(Low-power mode) 此模式用于在 USB 挂起状态下降低功耗。在此模式下，除了外接上拉电阻的供电，其他的静态功耗都被关闭，系统时钟将会停止或者降低到一定的频率来减少耗电。USB 总线上的活动(唤醒事件)将会复位此位(软件也可以复位此位)。 0：非低功耗模式 1：低功耗模式
位 1	**PDWN**：断电模式(Power down) 此模式用于彻底关闭 USB 模块。当此位被置位时，不能使用 USB 模块。 0：退出断电模式 1：进入断电模式
位 0	**FRES**：强制 USB 复位(Force USB Reset) 0：清除 USB 复位信号 1：对 USB 模块强制复位，类似于 USB 总线上的复位信号。USB 模块将一直保持在复位状态下直到软件清除此位。如果 USB 复位中断被使能，将产生一个复位中断。

USB 中断状态寄存器(USB_ISTR)

地址偏移：0x44

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR	PMA OVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	保留			DIR	EP_ID[3:0]			
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	res			r	r	r	r	r

此寄存器包含所有中断源的状态信息，以供应用程序确认产生中断请求的事件。

寄存器的高 8 位各表示一个中断源。当相关事件发生时，这些位被硬件置位，如果 USB_CNTR 寄存器上的相应位也被置位，则会产生相应的中断。中断服务程序需要检查每个位，在执行必要的操作后必需清除相应的状态位，不然中断信号线一直保持为高，同样的中断会再次被触发。如果同时多个中断标志被设置，也只会产生一个中断。

应用程序可以使用不同的方式处理传输完成中断，以减少中断响应的延迟时间。端点在成功完成一次传输后，CTR 位会被硬件置起，如果 USB_CNTR 上的相应位也被设置的话，就会产生中断。与端点相关的中断标志和 USB_CNTR 寄存器的 CTRM 位无关。这两个中断标志位将一直保持有效，直到应用程序清除了 USB_EPnR 寄存器中的相关中断挂起位(CTR 位是个只读位)。

USB 模块有两路中断请求源：

高优先级的 USB IRQ：用于高优先级的端点(同步和双缓冲批量端点)的中断请求，并且该中断不能被屏蔽。

低优先级 USB IRQ：用于其他中断事件，可以是低优先级的不可屏蔽中断，也可以是由 USB_ISTR 寄存器的高 8 位标识的可屏蔽中断。

对于端点产生的中断，应用程序可以通过 DIR 寄存器和 EP_ID 只读位来识别中断请求由哪个端点产生，并调用相应的中断服务程序。

用户在处理同时发生的多个中断事件时，可以在中断服务程序里检查 USB_ISTR 寄存器各个位的顺序来确定这些事件的优先级。在处理完相应位的中断后需要清零该中断标志。完成一次中断服务后，另一中断请求将会产生，用以请求处理剩下的中断事件。

为了避免意外清零某些位，建议使用加载指令，对所有不需改变的位写'1'，对需要清除的位写'0'。对于该寄存器，不建议使用读出－修改－写入的流程，因为在读写操作之间，硬件可能需要设置某些位，而这些位会在写入时被清零。

下面详细描述每个位：

位 15	**CTR**：正确的传输(Correct transfer) 此位在端点正确完成一次数据传输后由硬件置位。应用程序可以通过 DIR 和 EP_ID 位来识别是哪个端点完成了正确的数据传输。 此位应用程序只读
位 14	**PMAOVR**：分组缓冲区溢出(Packet memory area over / underrun) 此位在微控制器长时间没有响应一个访问 USB 分组缓冲区请求时由硬件置位。USB 模块通常在以下情况时置位该位：在接收过程中一个 ACK 握手分组没有被发送，或者在发送过程中发生了比特填充错误，在以上两种情况下主机都会要求数据重传。在正常的数据传输中不会产生 PMAOVR 中断。由于失败的传输都将由主机发起重传，应用程序就可以在这个中断的服务程序中加速设备的其他操作，并准备重传。但这个中断不会在同步传输中产生(同步传输不支持重传)因此数据可能会丢失。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 13	**ERR**：出错(Error) 在下列错误发生时硬件会置位此位。 NANS：无应答。主机的应答超时。 CRC：循环冗余校验码错误。数据或令牌分组中的 CRC 校验出错。

	BST：位填充错误。PID，数据或 CRC 中检测出位填充错误。

	FVIO：帧格式错误。收到非标准帧(如 EOP 出现在错误的时刻，错误的令牌等)。 USB 应用程序通常可以忽略这些错误，因为 USB 模块和主机在发生错误时都会启动重传机制。此位产生的中断可以用于应用程序的开发阶段，可以用来监测 USB 总线的传输质量，标识用户可能发生的错误(连接线松，环境干扰严重，USB 线损坏等)。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 12	**WKUP**：唤醒请求(Wakeup) 当 USB 模块处于挂起状态时，如果检测到唤醒信号，此位将由硬件置位。此时 CTLR 寄存器的 LP_MODE 位将被清零，同时 USB_WAKEUP 被激活，通知设备的其他部分(如唤醒单元)将开始唤醒过程。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 11	**SUSP**：挂起模块请求(Suspend mode request) 此位在 USB 线上超过 3ms 没有信号传输时由硬件置位，用以指示一个来自 USB 总线的挂起请求。USB 复位后硬件立即使能对挂起信号的检测，但在挂起模式下(FSUSP＝1)硬件不会再检测挂起信号直到唤醒过程结束。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 10	**RESET**：USB 复位请求(USB reset request) 此位在 USB 模块检测到 USB 复位信号输入时由硬件置位。此时 USB 模块将复位内部协议状态机，并在中断使能的情况下触发复位中断来响应复位信号。USB 模块的发送和接收部分将被禁止，直到此位被清除。所有的配置寄存器不会被复位，除非应用程序对他们清零。这用来保证在复位后 USB 传输还可以立即正确执行。但设备的地址和端点寄存器会被 USB 复位所复位。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 9	**SOF**：帧首标志(Start of frame) 此位在 USB 模块检测到总线上的 SOF 分组时由硬件置位，标志一个新的 USB 帧的开始。中断服务程序可以通过检测 SOF 事件来完成与主机的 1ms 同步，并正确读出寄存器在收到 SOF 分组时的更新内容(此功能在同步传输时非常有意义)。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 8	**ESOF**：期望帧首标识位(Expected start of frame) 此位在 USB 模块未收到期望的 SOF 分组时由硬件置位。主机应该每毫秒都发送 SOF 分组，但如果 USB 模块没有收到，挂起定时器将触发此中断。如果连续发生 3 次 ESOF 中断，也就是连续 3 次未收到 SOF 分组，将产生 SUSP 中断。即使在挂起定时器未被锁定时发生 SOF 分组丢失，此位也会被置位。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 7:5	保留
位 4	**DIR**：传输方向(Direction of transaction) 此位在完成数据传输产生中断后由硬件根据传输方向写入。 如果 DIR＝0，相应端点的 CTR_TX 位被置位，标志一个 IN 分组(数据从 USB 模块传输到 PC 主机）的传输完成。 如果 DIR＝1，相应端点的 CTR_RX 位被置位，标志一个 OUT 分组(数据从 PC 主机传输到 USB 块)的传输完成。如果 CTR_TX 位同时也被置位，就标志同时存在挂起的 OUT 分组和 IN 分组。 应用程序可以利用该信息访问 USB_EPnR 位对应的操作，它表示挂起中断传输方向的信息。 该位为只读
位 3:0	**EP_ID[3:0]**：端点 ID (Endpoint Identifier) 此位在 USB 模块完成数据传输产生中断后由硬件根据请求中断的端点号写入。如果同时有多个端点的请求中断，硬件写入优先级最高的端点号。端点的优先级按以下方法定义：同步端点和双缓冲批量端点具有高优先级，其他的端点为低优先级。如果多个同优先级的端点请求中断，则根据端点号来确定优先级，即端点 0 具有最高优先级，端点号越小，优先级越高。应用程序可以通过上述的优先级策略顺序处理端点的中断请求。 该位为只读。

USB 帧编号寄存器(USB_FNR)

地址偏移：0x48

复位值：0x0XXX，X 代表未定义数值

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]										
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 15	**RXDP**：D+状态位(Receive data + line status) 此位用于观察 USB D+数据线的状态，可在挂起状态下检测唤醒条件的出现。
位 14	**RXDM**：D-状态位(Receive data - line status) 此位用于观察 USB D-数据线的状态，可在挂起状态下检测唤醒条件的出现。
位 13	**LCK**：锁定位(Locked) USB 模块在复位或唤醒序列结束后会检测 SOF 分组，如果连续检测到至少 2 个 SOF 分组，则硬件会置位此位。此位一旦锁定，帧计数器将停止计数，一直等到 USB 模块复位或总线挂起时再恢复计数。
位 12:11	**LSOF[1:0]**：帧首丢失标志位(Lost SOF) 当 ESOF 事件发生时，硬件会将丢失的 SOF 分组的数目写入此位。如果再次收到 SOF 分组，引脚会清除此位。

位 10:0	**FN[10:0]**：帧编号(Frame number) 此部分记录了最新收到的 SOF 分组中的 11 位帧编号。主机每发送一个帧，帧编号都会自加，这对于同步传输非常有意义。此部分发生 SOF 中断时更新。

USB 设备地址寄存器(USB_DADDR)

地址偏移：0x4C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								EF	ADD[6:0]						
								rw	rw	rw	rw	rw	rw	rw	rw

位 7	**EF**：USB 模块使能位(Enable function) 此位在需要使能 USB 模块时由应用程序置位。如果此位为 0，USB 模块将停止工作，忽略所有寄存器的设置，不响应任何 USB 通信。
位 6:0	**ADD[6:0]**：设备地址(evice address) 此位记录了 USB 主机在枚举过程中为 USB 设备分配的地址值。该地址值和端点地址(EA)必需和 USB 令牌分组中的地址信息匹配，才能在指定的端点进行正确的 USB 传输。

USB 分组缓冲区描述表地址寄存器(USB_BTABLE)

地址偏移：0x50

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BTABLE[15:3]													保留		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res		

位 15:3	**BTABLE[15:3]**：缓冲表(Buffer table) 此位记录分组缓冲区描述表的起始地址。分组缓冲区描述表用来指示每个端点的分组缓冲区地址和大小，按 8 字节对齐(即最低 3 位为 000)。每次传输开始时，USB 模块读取相应端点所对应的分组缓冲区描述表获得缓冲区地址和大小信息。
位 2:0	保留位，由硬件置为 0

17.5.2 端点寄存器

端点寄存器的数量由 USB 模块所支持的端点数目决定。USB 模块最多支持 8 个双向端点。每个USB 设备必须支持一个控制端点，控制端点的地址(EA 位)必需为 0。不同的端点必需使用不同的端点号，否则端点的状态不定。每个端点都有与之对应的 USB_EPnR 寄存器，用于存储该端点的各种状态信息。

USB 端点 n 寄存器(USB_EPnR)，n=[0..7]

地址偏移：0x00 至 0x1C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR_RX	DTOG_RX	STAT_RX[1:0]		SETUP	EP TYPE[1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX[1:0]		EA[3:0]			
rc_w0	t	t	t	r	rw	rw	rw	rc_w0	t	t	t	rw	rw	rw	rw

当 USB 模块收到 USB 总线复位信号，或 CTLR 寄存器的 FRES 位置位时，USB 模块将会复位。该寄存器除了 CTR_RX 和 CTR_TX 位保持不变以处理紧随的 USB 传输外，其他位都被复位。每个端点对应一个 USB_EPnR 寄存器，其中 n 为端点地址，即端点 ID 号。

对于此类寄存器应避免执行读出－修改－写入操作，因为在读和写操作之间，硬件可能会设置某些位，而这些位又会在写入时被修改，导致应用程序错过相应的操作。因此，这些位都有一个写入无效的值，建议用 Load 指令修改这些寄存器，以免应用程序修改了不需要修改的位。

位 15	**CTR_RX**：正确接收标志位(Correct Transfer for reception) 此位在正确接收到 OUT 或 SETUP 分组时由硬件置位，应用程序只能对此位清零。如果 CTRM 位已置位，相应的中断会产生。收到的是 OUT 分组还是 SETUP 分组可以通过下面描述的 SETUP 位确定。以 NAK 或 STALL 结束的分组和出错的传输不会导致此位置位，因为没有真正传输数据。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 14	**DTOG_RX**：用于数据接收的数据翻转位(Data Toggle, for reception transfers) 对于非同步端点，此位由硬件设置，用于标记希望接收的下一个数据分组的 Toggle 位(0=DATA0，1=DATA1)。在接收到 PID(分组 ID)正确的数据分组之后，USB 模块发送 ACK 握手分组，并翻转此位。对于控制端点，硬件在收到 SETUP 分组后清除此位。 对于双缓冲端点，此位还用于支持双缓冲区的交换(请参考 18.4.3 双缓冲端点)。 对于同步端点，由于仅发送 DATA0，因此此位仅用于支持双缓冲区的交换(请参考 18.4.4 同步传输)而不需进行翻转。同步传输不需要握手分组，因此硬件在收到数据分组后立即设置此位。 应用程序可以对此位进行初始化(对于非控制端点，初始化是必需的)，或者翻转此位用于特殊用途。 此位应用程序可读可写，但写 0 无效，写 1 可以翻转此位。
位 13:12	**STAT_RX[1:0]**：用于数据接收的状态位(Status bits, for reception transfers) 此位用于指示端点当前的状态，表 67 列出了端点的所有状态。当一次正确的 OUT 或 SETUP 数据传输完成后(CTR_RX=1)，硬件会自动设置此位为 NAK 状态，使应用程序有足够的时间在处理完当前传输的数据后，响应下一个数据分组。 对于双缓冲批量端点，由于使用特殊的传输流量控制策略，因此根据使用的缓冲区状态控制传输状态(请参考 18.4.3 双缓冲端点)。 对于同步端点，由于端点状态只能是有效或禁用，因此硬件不会在正确的传输之后设置此位。如果应用程序将此位设为 STALL 或者 NAK，USB 模块响应的操作是未定义的。 此位应用程序可读可写，但写 0 无效，写 1 翻转此位。

位 11	**SETUP**：SETUP 分组传输完成标志位(Setup transaction completed) 此位在 USB 模块收到一个正确的 SETUP 分组后由硬件置位，只有控制端点才使用此位。在接收完成后(CTR_RX=1)，应用程序需要检测此位以判断完成的传输是否是 SETUP 分组。为了防止中断服务程序在处理 SETUP 分组时下一个令牌分组修改了此位，只有 CTR_RX 为 0 时，此位才可以被修改，CTR_RX 为 1 时不能修改。 此位应用程序只读。
位 10:9	**EP_TPYE[1:0]**：端点类型位(Endpoint type) 此位用于指示端点当前的类型，所有的端点类型都在表 68 中列出。所有的 USB 设备都必需包含一个地址为 0 的控制端点，如果需要可以有其他地址的控制端点。只有控制端点才会有 SETUP 传输，其他类型的端点无视此类传输。SETUP 传输不能以 NAK 或 STALL 分组响应，如果控制端点在收到 SETUP 分组时处于 NAK 状态，USB 模块将不响应分组，就会出现接收错误。如果控制端点处于 STALL 状态，SETUP 分组会被正确接收，数据会被正确传输，并产生一个正确传输完成的中断。控制端点的 OUT 分组安装普通端点的方式处理。 批量端点和中断端点的处理方式非常类似，仅在对 EP_KIND 位的处理上有差别。 同步端点的用法请参考 18.4.4 同步传输。
位 8	**EP_KIND**：端点特殊类型位(Endpoint kind) 此位的需要和 EP_TYPE 位配合使用，具体的定义请参考表 69。 DBL_BUF：应用程序设置此位能使能批量端点的双缓冲功能。详见 18.4.3 双缓冲端点。 STATUS_OUT：应用程序设置此位表示 USB 设备期望主机发送一个状态数据分组，此时，设备对于任何长度不为 0 的数据分组都响应 STALL 分组。此功能仅用于控制端点，有利于提供应用程序对于协议层错误的检测。如果 STATUS_OUT 位被清除，OUT 分组可以包含任意长度的数据。
位 7	**CTR_TX**：正确发送标志位(Correct transfer for transmission) 此位由硬件在一个正确的 IN 分组传输完成后置位。如果 CTRM 位已被置位，会产生相应的中断。应用程序需要在处理完该事件后清除此位。在 IN 分组结束时，如果主机响应 NAK 或 STALL 则此位不会被置位，因为数据传输没有成功。 此位应用程序可读可写，但写 0 有效，写 1 无效。

位 6	**DTOG_RX**：发送数据翻转位(Data Toggle, for transmission transfers) 对于非同步端点，此位用于指示下一个要传输的数据分组的 Toggle 位(0＝DATA0，1=DATA1)。在一个成功传输的数据分组后，如果 USB 模块接收到主机发送的 ACK 分组，就会翻转此位。 对于控制端点，USB 模块会在收到正确的 SETUP PID 后置位此位。 对于双缓冲端点，此位还可用于支持分组缓冲区交换(请参考 18.4.3 双缓冲端点)。 对于同步端点，由于只传送 DATA0，因此该位只用于支持分组缓冲区交换(请参考 18.4.4 同步传输)。由于同步传输不需要握手分组，因此硬件在接收到数据分组后即设置该位。 应用程序可以初始化该位(对于非控制端点，初始化此位时必需的)，也可以设置该位用于特殊用途。 此位应用程序可读可写，但写 0 无效，写 1 翻转此位。
位 5:4	**STAT_TX[1:0]**：用于发送数据的状态位(Status bits, for transmission transfers) 此位用于标识端点的当前状态，表 70 列出了所有的状态。应用程序可以翻转这些位来初始化状态信息。在正确完成一次 IN 分组的传输后(CTR_TX=1)，硬件会自动设置此位为 NAK 状态，保证应用程序有足够的时间准备好数据响应后续的数据传输。 对于双缓冲批量端点，由于使用特殊的传输流量控制策略，是根据缓冲区的状态控制传输的状态的(请参考 18.4.3 双缓冲端点)。 对于同步端点，由于端点的状态只能是有效或禁用，因此硬件不会在数据传输结束时改变端点的状态。如果应用程序将此位设为 STALL 或者 NAK，则 USB 模块后续的操作是未定义的。 此位应用程序可读可写，但写 0 无效，写 1 翻转此位。
位 3:0	**EA[3:0]**：端点地址(Endpoint address) 应用程序必需设置此 4 位，在使能一个端点前为它定义一个地址。

表 67 接收状态编码

STAT_RX[1:0]	描述
00	**DISABLED**：端点忽略所有的接收请求。
01	**STALL**：端点以 STALL 分组响应所有的接收请求。
10	**NAK**：端点以 NAK 分组响应所有的接收请求。
11	**VALID**：端点可用于接收。

表 68 端点类型编码

EP_TYPE[1:0]	描述
00	**BULK**：批量端点
01	**CONTROL**：控制端点
10	**ISO**：同步端点
11	**INTERRUPT**：中断端点

表 69 端点特殊类型定义

EP_TYPE[1:0]		EP_KIND 意义
00	BULK	DBL_BUF：双缓冲端点
01	CONTROL	STATUS_OUT
10	ISO	未使用
11	INTERRUPT	未使用

表 70 发送状态编码

STAT_RX[1:0]	描述
00	DISABLED：端点忽略所有的发送请求。
01	STALL：端点以 STALL 分组响应所有的发送请求。
10	NAK：端点以 NAK 分组响应所有的发送请求。
11	VALID：端点可用于发送。

17.5.3 缓冲区描述表

虽然缓冲区描述表位于分组缓冲区内，但仍可将它看作是特殊的寄存器，用以配置 USB 模块和微控制器内核共享的分组缓冲区的地址和大小。由于 APB1 总线按 32 位寻址，所以所有的分组缓冲区地址都使用 32 位对齐的地址，而不是 USB_BTABLE 寄存器和缓冲区描述表所使用的地址。

以下介绍两种地址表示方式：一种是应用程序访问分组缓冲区时使用的，另一种是相对于 USB 模块的本地地址。供应用程序使用的分组缓冲区地址需要乘以 2 才能得到缓冲区在微控制器中的真正地址。分组缓冲区的首地址为 0x4000 6000。下面将描述与 USB_EPnR 寄存器相关的缓冲区描述表。完整的分组缓冲区的说明和缓冲区描述表的用法请参考 18.4.2 节。

发送缓冲区地址寄存器 n(USB_ADDRn_TX)

地址偏移：[USB_BTABLE] + n×16

USB 本地地址：[USB_BTABLE] + n×8

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRn_TX[15:1]															-
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	-

位 15:1	**ADDRn_TX[15:1]**：发送缓冲区地址(Transmission buffer address) 此位记录了收到下一个 IN 分组时，需要发送的数据所在的缓冲区起始地址。
位 0	因为分组缓冲区的地址必须按字对齐，所以此位必须为'0'。

发送数据字节数寄存器 n(USB_COUNTn_TX)

地址偏移：[USB_BTABLE] + n×16 + 4

USB 本地地址：[USB_BTABLE] + n×8 + 2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-						COUNTn_TX[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:10	由于 USB 模块支持的最大数据分组为 1023 个字节，所以 USB 模块忽略这些位。
位 9:0	**COUNTn_TX[9:0]**：发送数据字节数(Transmission byte count) 此位记录了收到下一个 IN 分组时要传输的数据字节数。

注： *双缓冲区和同步 IN 端点有两个 USB_COUNTn_TX 寄存器：分别为 USB_COUNTn_TX_1 和 USB_COUNTn_TX_0，内容如下：*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
-						COUNTn_TX_1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-						COUNTn_TX_0[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

接收缓冲区地址寄存器 n(USB_ADDRn_RX)

地址偏移：[USB_BTABLE] + n×16 + 8

USB 本地地址：[USB_BTABLE] + n×8 + 4

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRn_RX[15:1]															-
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	-

位 15:1	**ADDRn_RX[15:1]**：接收缓冲区地址(Reception buffer address) 此位记录了收到下一个 OUT 或者 SETUP 分组时，用于保存数据的缓冲区起始地址。
位 0	因为分组缓冲区的地址按字对齐，所以此位必需为'0'。

接收数据字节数寄存器 n(USB_COUNTn_RX)

地址偏移：[USB_BTABLE] + n×16 + 12

USB 本地地址：[USB_BTABLE] + n×8 + 6

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLSIZE	NUM_BLOCK[4:0]					COUNTn_RX[9:0]									
rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r	r	r

该寄存器用于存放接收分组时需要使用到的两个参数。最高的标志位定义了接收分组缓冲区的大小，以便 USB 模块检测缓冲区的溢出。低标志位则用于 USB 模块记录实际接收到的字节数。由于有效位数的限制，缓冲区的大小由分配到的存储区块数表示，而存储区块的大小则由所需的缓冲区大小决定。缓冲区的大小在设备枚举过程中定义，由端点描述符的参数 maxPacketSize 表述。(具体信息请参考"USB 2.0 协议规范")

位 15	**BL_SIZE**：存储区块的大小(Block size) 此位用于定义决定缓冲区大小的存储区块的大小。 如果 BL_SIZE=0，存储区块的大小为 2 字节，因此能分配的分组缓冲区的大小范围为 2－62 个字节。 如果 BL_SIZE=1，存储区块的大小为 32 字节，因此能分配的分组缓冲区的大小范围为 32－1024 字节，符合 USB 协议定义的最大分组长度限制。
位 14:10	**NUM_BLOCK[4:0]**：存储区块的数目(Number of blocks) 此位用以记录分配的存储区块的数目，从而决定最终使用的分组缓冲区的大小。具体请参考表71
位 9:0	**COUNTn_RX[9:0]**：接收到的字节数(Reception byte count) 此位由 USB 模块写入，用以记录端点收到的和 USB_EPnR 相关的最新的 OUT 或 SETUP 分组的实际字节数。

注：双缓冲区和同步 IN 端点有两个 USB_COUNTn_RX 寄存器：分别为 USB_COUNTn_RX_1 和 USB_COUNTn_RX_0，内容如下：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLSIZE_1	NUM_BLOCK_1[4:0]					COUNTn_RX_1[9:0]									
rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLSIZE_0	NUM_BLOCK_0[4:0]					COUNTn_RX_0[9:0]									
rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r	r	r

表 71 分组缓冲区大小的定义

NUM_BLOCK[4:0]的值	BL_SIZE=0 时的分组缓冲区大小	当 BL_SIZE=1 时的 分组缓冲区大
00000	不允许使用	32 字节
00001	2 字节	64 字节
00010	4 字节	96 字节
00011	6 字节	128 字节
…	…	…
01111	30 字节	512 字节
10000	32 字节	保留
10001	34 字节	保留
10010	36 字节	保留
…	…	…
11110	60 字节	保留
11111	62 字节	保留

17.5.4 USB 寄存器映像

表 72 USB 寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00h	USB_EP0R	保留																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04h	USB_EP1R																	CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08h	USB_EP2R																	CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0Ch	USB_EP3R																	CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10h	USB_EP4R																	CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14h	USB_EP5R																	CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18h	USB_EP6R																	CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1Ch	USB_EP7R																	CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20h~ 3Fh	保留																																
40h	USB_CNTR	保留																CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	保留			RESUME	FSUSP	LPMODE	PDWN	FRES
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
44h	USB_ISTR	保留																CTR	PMAOVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	保留			DIR	EP_ID [3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48h	USB_FNR	保留																RXDP	RXDM	LCK	LSOF [1:0]		FN[10:0]										
	复位值																	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X
4Ch	USB_DADDR	保留																保留								EF	ADD[6:0]						
	复位值																										0	0	0	0	0	0	0
50h	USB_BTABLE	保留																BTABLE[15:3]														保留	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

关于寄存器的起始地址，请参见表 1。

18 控制器局域网（bxCAN）

18.1 bxCAN 简介

bxCAN 是基本扩展 CAN(Basic Extended CAN)的缩写，它支持 CAN 协议 2.0A 和 2.0B。它的设计 目标是，以最小的 CPU 负荷来高效处理大量收到的报文。它也支持报文发送的优先级要求(优先 级特性可软件配置)。

对于安全紧要的应用，bxCAN 提供所有支持时间触发通信模式所需的硬件功能。

18.2 bxCAN 主要特点

- 支持 CAN 协议 2.0A 和 2.0B 主动模式
- 波特率最高可达 1 兆位/秒
- 支持时间触发通信功能

发送

- 3 个发送邮箱
- 发送报文的优先级特性可软件配置
- 记录发送 SOF 时刻的时间戳

接收

- 3 级深度的 2 个接收 FIFO
- 可变的过滤器组：
  - 其它 HK32F103xx 系列产品中有 14 个过滤器组
- 标识符列表
- FIFO 溢出处理方式可配置
- 记录接收 SOF 时刻的时间戳

时间触发通信模式

- 禁止自动重传模式
- 16 位自由运行定时器
- 可在最后 2 个数据字节发送时间戳

管理

- 中断可屏蔽
- 邮箱占用单独 1 块地址空间，便于提高软件效率

CAN

- CAN：是主 bxCAN，它负责管理在从 bxCAN 和 512 字节的 SRAM 存储器之间的通信
- bxCAN 模块使用 512 字节的 SRAM 存储器(见图 140)

注： *USB 和 CAN 共用一个专用的 512 字节的 SRAM 存储器用于数据的发送和接收，因此不能同时使用 USB 和 CAN(共享的 SRAM 被 USB 和 CAN 模块互斥地访问)。USB 和 CAN 可以同时用于一个应用中但不能在同一个时间使用。*

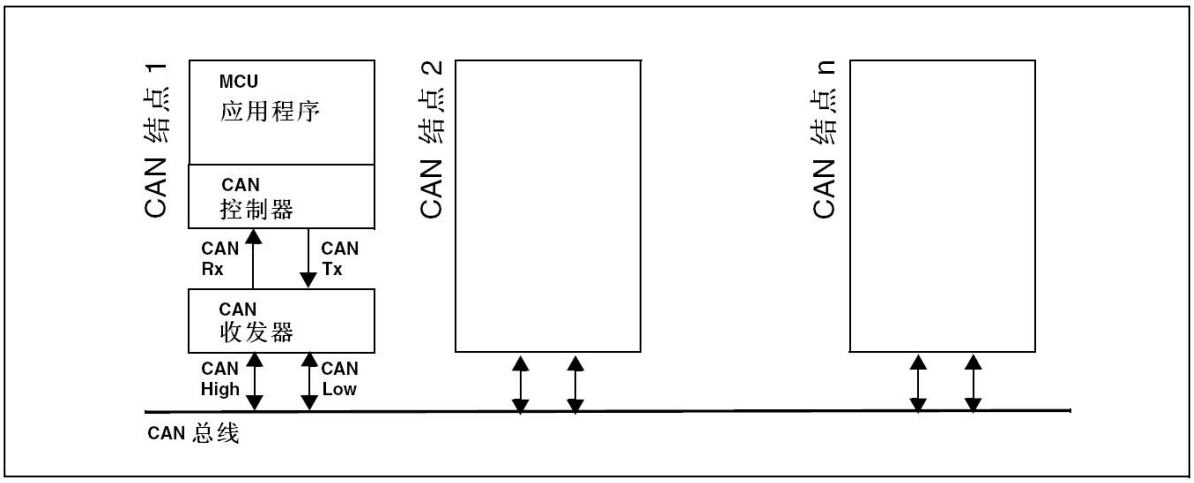
18.3 bxCAN 总体描述

在当今的 CAN 应用中，CAN 网络的节点在不断增加，并且多个 CAN 常常通过网关连接起来，因 此整个 CAN 网中的报文数量(每个节点都需要处理)急剧增加。除了应用层报文外，网络管理和诊断报文也被引入。

- 需要一个增强的过滤机制来处理各种类型的报文此外，应用层任务需要更多 CPU 时间，因此报文接收所需的实时响应程度需要减轻。
- 接收 FIFO 的方案允许，CPU 花很长时间处理应用层任务而不会丢失报文。

构筑在底层 CAN 驱动程序上的高层协议软件，要求跟 CAN 控制器之间有高效的接口。

图 139 CAN 网拓扑结构



18.3.1 CAN 2.0B 主动内核

bxCAN 模块可以完全自动地接收和发送 CAN 报文；且完全支持标准标识符(11 位)和扩展标识符(29 位)。

18.3.2 控制、状态和配置寄存器

应用程序通过这些寄存器，可以：

- 配置 CAN 参数，如波特率
- 请求发送报文
- 处理报文接收
- 管理中断
- 获取诊断信息

18.3.3 发送邮箱

共有 3 个发送邮箱供软件来发送报文。发送调度器根据优先级决定哪个邮箱的报文先被发送。

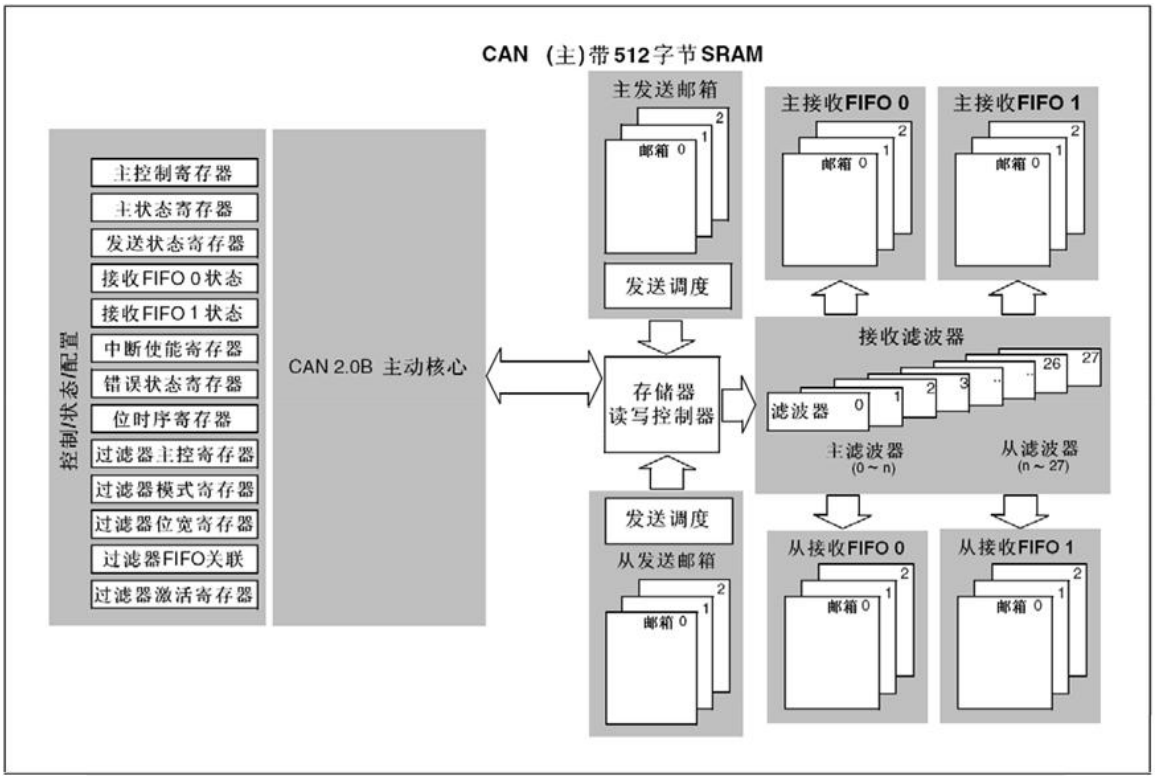
18.3.4 接收过滤器

bxCAN 提供 14 个位宽可变/可配置的标识符过滤器组，软件通过对它们编程，从而在引脚收到的报文中选择它需要的报文，而把其它报文丢弃掉。

接收 FIFO

共有 2 个接收 FIFO，每个 FIFO 都可以存放 3 个完整的报文。它们完全由硬件来管理。

图 140 CAN 框图



18.4 bxCAN 工作模式

bxCAN 有 3 个主要的工作模式：**初始化、正常和睡眠模式**。

在硬件复位后，bxCAN 工作在睡眠 模式以节省电能，同时 CANTX 引脚的内部上拉电阻被激活。软件通过对 CAN_MCR 寄存器的INRQ 或 SLEEP 位置'1'，可以请求 bxCAN 进入**初始化**或**睡眠模式**。一旦进入了**初始化**或**睡眠模式**，bxCAN 就对 CAN_MSR 寄存器的 INAK 或 SLAK 位置'1'来进行确认，同时内部上拉电阻被禁用。当 INAK 和 SLAK 位都为'0'时，bxCAN 就处于**正常模式**。在进入**正常模式**前，bxCAN 必须跟 CAN 总线取得**同步**；为取得同步，bxCAN 要等待 CAN 总线达到空闲状态，即在 CANRX 引脚上监测到 11 个连续的隐性位。

18.4.1 初始化模式

软件初始化应该在硬件处于初始化模式时进行。设置 CAN_MCR 寄存器的 INRQ 位为'1'，请求 bxCAN 进入初始化模式，然后等待硬件对 CAN_MSR 寄存器的 INAK 位置'1'来进行确认。

清除 CAN_MCR 寄存器的 INRQ 位为'0'，请求 bxCAN 退出初始化模式，当硬件对 CAN_MSR 寄存器的 INAK 位清'0'就确认了初始化模式的退出。

当 bxCAN 处于初始化模式时，禁止报文的接收和发送，并且 CANTX 引脚输出隐性位(高电平)。初始化模式的进入，不会改变配置寄存器。

软件对 bxCAN 的初始化，至少包括位时间特性(CAN_BTR)和控制(CAN_MCR)这 2 个寄存器。在对 bxCAN 的过滤器组(模式、位宽、FIFO 关联、激活和过滤器值)进行初始化前，软件要对 CAN_FMR 寄存器的 FINIT 位设置'1'。对过滤器的初始化可以在非初始化模式下进行。

注：　当 FINIT=1 时，报文的接收被禁止。

可以先对过滤器激活位清’0’(在 CAN_FA1R 中)，然后修改相应过滤器的值。 如果过滤器组没有使用，那么就应该让它处于非激活状态(保持其 FACT 位为清’0’状态)。

18.4.2 正常模式

在初始化完成后，软件应该让硬件进入正常模式，以便正常接收和发送报文。软件可以通过对 CAN_MCR 寄存器的 INRQ 位清’0’，来请求从初始化模式进入正常模式，然后要等待硬件对CAN_MSR 寄存器的 INAK 位置’1’的确认。在跟 CAN 总线取得同步，即在 CANRX 引脚上监测到11 个连续的隐性位(等效于总线空闲)后，bxCAN 才能正常接收和发送报文。

不需要在初始化模式下进行过滤器初值的设置，但必须在它处在非激活状态下完成(相应的 FACT 位为 0)。而过滤器的位宽和模式的设置，则必须在初始化模式中进入正常模式前完成。

18.4.3 睡眠模式(低功耗)

bxCAN 可工作在低功耗的睡眠模式。软件通过对 CAN_MCR 寄存器的 SLEEP 位置’1’，来请求进 入这一模式。在该模式下，bxCAN 的时钟停止了，但软件仍然可以访问邮箱寄存器。

当 bxCAN 处于睡眠模式，软件必须对 CAN_MCR 寄存器的 INRQ 位置’1’并且同时对 SLEEP 位清’0’，才能进入初始化模式。

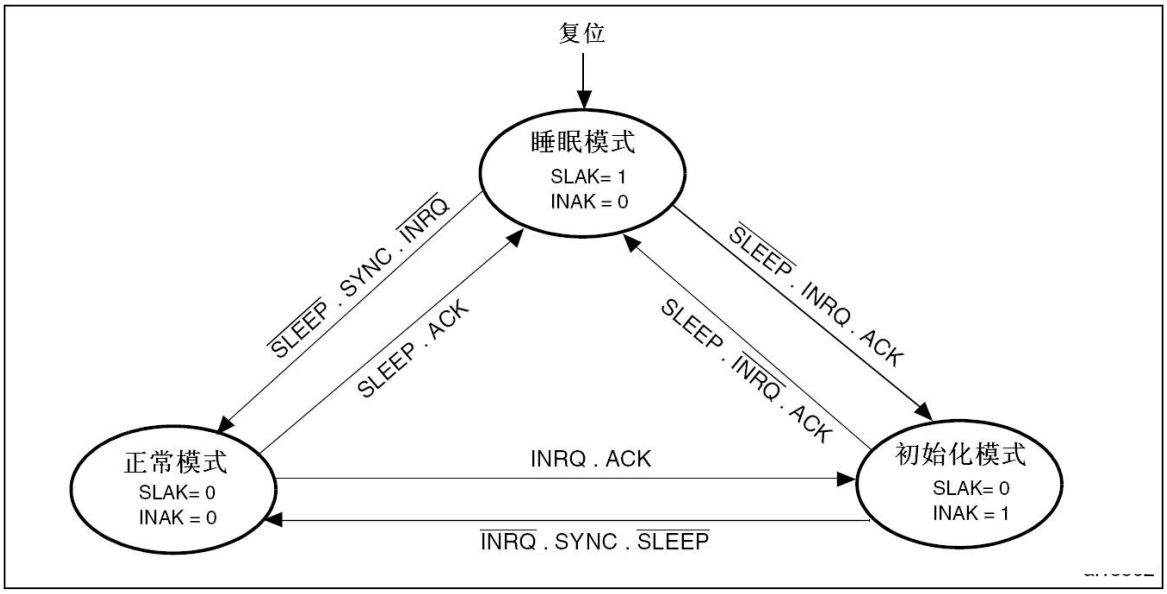
有 2 种方式可以唤醒(退出睡眠模式)bxCAN：通过软件对 SLEEP 位清’1’，或硬件检测到 CAN 总线的活动。

如果 CAN_MCR 寄存器的 AWUM 位为’1’，一旦检测到 CAN 总线的活动，硬件就自动对 SLEEP 位清’0’来唤醒 bxCAN。如果 CAN_MCR 寄存器的 AWUM 位为’0’，软件必须在唤醒中断里对 SLEEP 位清’0’才能退出睡眠状态。

注： *如果唤醒中断被允许(CAN_IER 寄存器的 WKUIE 位为’1’)，那么一旦检测到 CAN 总线活动就会产生唤醒中断，而不管硬件是否会自动唤醒 bxCAN。*

在对 SLEEP 位清’0’后，睡眠模式的退出必须与 CAN 总线同步，请参考图 141：bxCAN 工作模式。当硬件对 SLAK 位清’0’时，就确认了睡眠模式的退出。

图 141 bxCAN 工作模式



注： *1 ACK = 硬件响应睡眠或初始化请求,而对 CAN_MSR 寄存器的 INAK 或 SLAK 位置 1 的状态*

2 SYNC = bxCAN 等待 CAN 总线变为空闲的状态，即在 CANRX 引脚上检测到连续的 11 个隐性位

18.5 测试模式

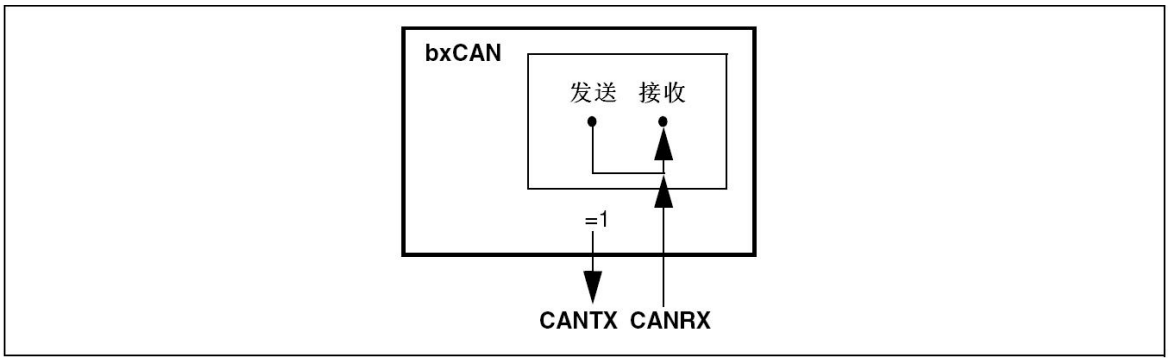
通过对 CAN_BTR 寄存器的 SILM 和/或 LBKM 位置'1'，来选择一种测试模式。只能在初始化模式下，修改这 2 位。在选择了一种测试模式后，软件需要对 CAN_MCR 寄存器的 INRQ 位清'0'，来真正进入测试模式。

18.5.1 静默模式

通过对 CAN_BTR 寄存器的 SILM 位置'1'，来选择静默模式。

在静默模式下，bxCAN 可以正常地接收数据帧和远程帧，但只能发出隐性位，而不能真正发送报文。如果 bxCAN 需要发出显性位(确认位、过载标志、主动错误标志)，那么这样的显性位在内部被接回来从而可以被 CAN 内核检测到，同时 CAN 总线不会受到影响而仍然维持在隐性位状态。因此，静默模式通常用于分析 CAN 总线的活动，而不会对总线造成影响－显性位(确认位、错误帧)不会真正发送到总线上。

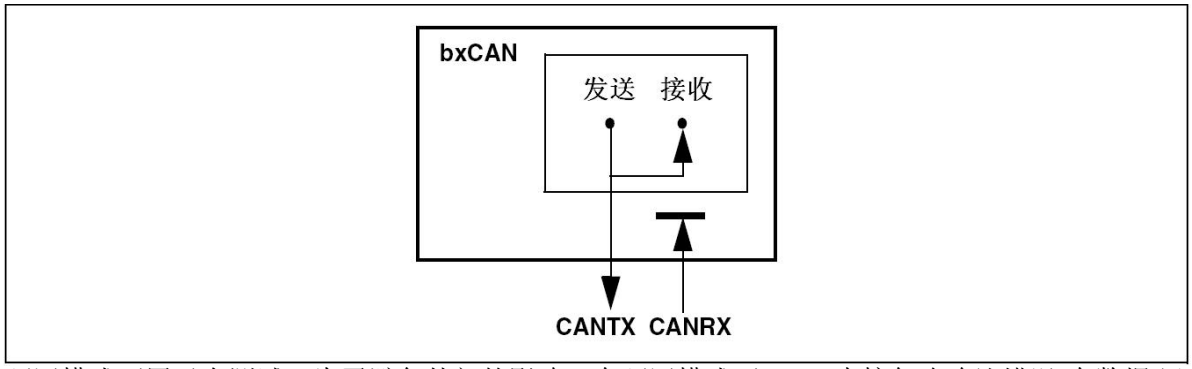
图 142 bxCAN 工作在静默模式



18.5.2 环回模式

通过对 CAN_BTR 寄存器的 LBKM 位置'1'，来选择环回模式。在环回模式下，bxCAN 把发送的报文当作接收的报文并保存(如果可以通过接收过滤)在接收邮箱里。

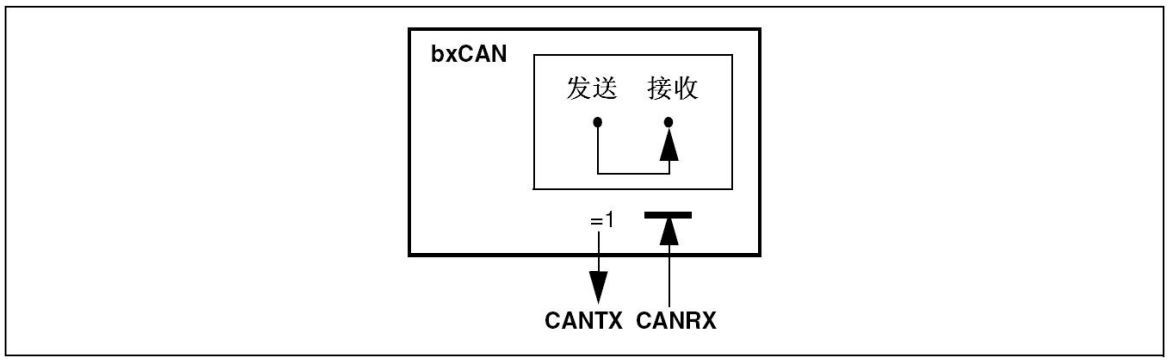
图 143 bxCAN 工作在环回模式



环回模式可用于自测试。为了避免外部的影响，在环回模式下 CAN 内核忽略确认错误(在数据/远程帧的确认位时刻，不检测是否有显性位)。在环回模式下，bxCAN 在内部把 Tx 输出回馈到 Rx 输入上，而完全忽略 CANRX 引脚的实际状态。发送的报文可以在 CANTX 引脚上检测到。

18.5.3 环回静默模式

图 144 bxCAN 工作在环回静默模式



通过对 CAN_BTR 寄存器的 LBKM 和 SILM 位同时置'1'，可以选择环回静默模式。该模式可用于“热自测试”，即可以像环回模式那样测试 bxCAN，但却不会影响 CANTX 和 CANRX 所连接的整个 CAN 系统。在环回静默模式下，CANRX 引脚与 CAN 总线断开，同时 CANTX 引脚被驱动到隐性位状态。

18.6 HK32F10xxx 处于调试模式时

当微控制器处于调试模式时，Cortex-M3 核心处于暂停状态，依据下述配置位的状态，bxCAN 可以继续正常工作或停止工作：

- 调试(DBG)模块中 CAN 的 DBG_CAN_STOP 位。详见第 25.15.2 节：支持定时器、看门狗、bxCAN 和 I^2C 的调试。
- CAN_MCR 中的 DBF 位。详见第 19.9.2 节：CAN 控制和状态寄存器。

18.7 bxCAN 功能描述

18.7.1 发送处理

发送报文的流程为：应用程序选择 1 个**空置**的发送邮箱；设置标识符，数据长度和待发送数据；然后对 CAN_TIxR 寄存器的 TXRQ 位置'1'，来请求发送。TXRQ 位置'1'后，邮箱就不再是空邮箱；而一旦邮箱不再为**空置**，软件对邮箱寄存器就不再有写的权限。TXRQ 位置 1 后，邮箱马上 进入**挂号**状态，并等待成为最高优先级的邮箱，参见*发送优先级*。一旦邮箱成为最高优先级的 邮箱，其状态就变为**预定**发送状态。一旦 CAN 总线进入空闲状态，预定发送邮箱中的报文就马 上被发送(进入**发送**状态)。一旦邮箱中的报文被成功发送后，它马上变为**空置**邮箱；硬件相应地对 CAN_TSR 寄存器的 RQCP 和 TXOK 位置 1，来表明一次成功发送。

如果发送失败，由于仲裁引起的就对 CAN_TSR 寄存器的 ALST 位置'1'，由于发送错误引起的就对 TERR 位置'1'。

发送优先级

由标识符决定

当有超过 1 个发送邮箱在挂号时，发送顺序由邮箱中报文的标识符决定。根据 CAN 协议，标识符数值最低的报文具有最高的优先级。如果标识符的值相等，那么邮箱号小的报文先被发送。

由发送请求次序决定

通过对 CAN_MCR 寄存器的 TXFP 位置'1'，可以把发送邮箱配置为发送 FIFO。在该模式下，发送的优先级由发送请求次序决定。

该模式对分段发送很有用。

中止

通过对 CAN_TSR 寄存器的 ABRQ 位置'1'，可以中止发送请求。邮箱如果处于**挂号**或**预定**状态，发送请求马上就被中止了。如果邮箱处于**发送**状态，那么中止请求可能导致 2 种结果。如果邮箱中的报文被成功发送，那么邮箱变为**空置**邮箱，并且 CAN_TSR 寄存器的 TXOK 位被硬件置'1'。如果邮箱中的报文发送失败了，那么邮箱变为**预定**状态，然后发送请求被中止，邮箱变为**空置**邮箱且 TXOK 位被硬件清'0'。因此如果邮箱处于**发送**状态，那么在发送操作结束后，邮箱都会变为**空置**邮箱。

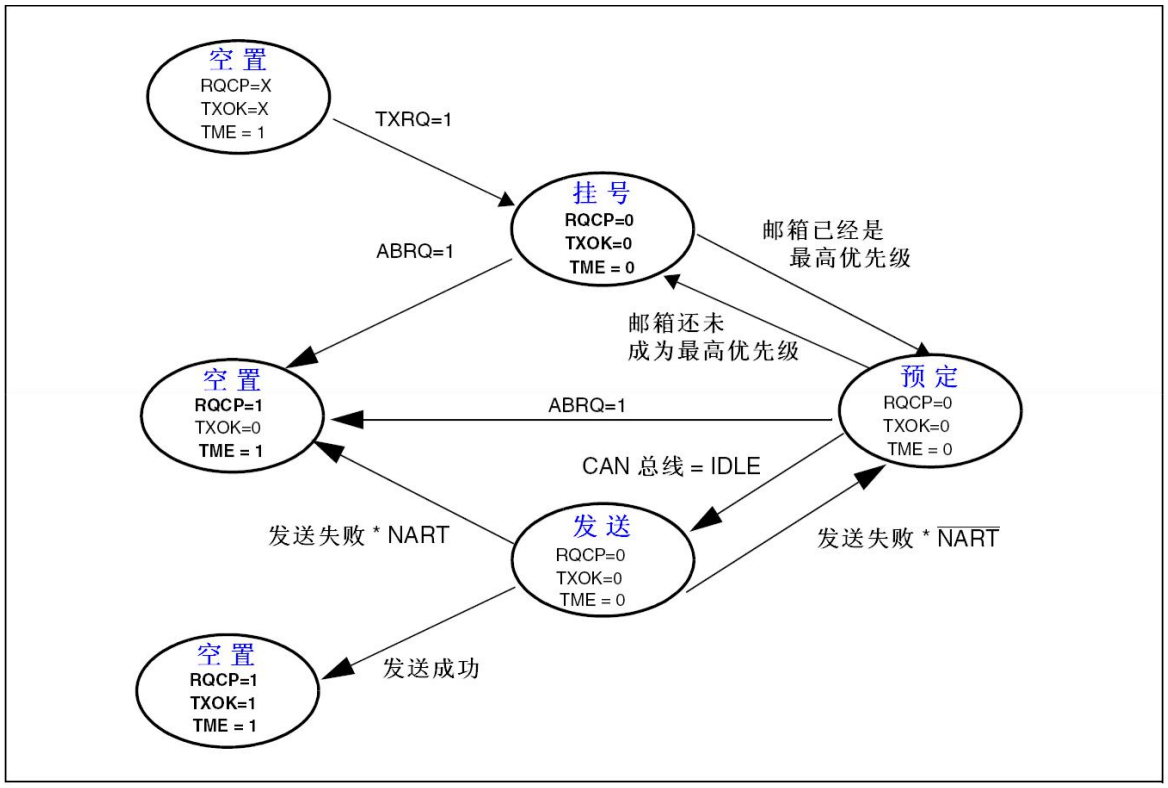
禁止自动重传模式

该模式主要用于满足 CAN 标准中，时间触发通信选项的需求。通过对 CAN_MCR 寄存器的 NART 位置'1'，来让硬件工作在该模式。

在该模式下，发送操作只会执行一次。如果发送操作失败了，不管是由于仲裁丢失或出错，硬件都不会再自动发送该报文。

在一次发送操作结束后，硬件认为发送请求已经完成，从而对 CAN_TSR 寄存器的 RQCP 位置'1'，同时发送的结果反映在 TXOK、ALST 和 TERR 位上。

图 145 发送邮箱状态



18.7.2 时间触发通信模式

在该模式下，CAN 硬件的内部定时器被激活，并且被用于产生(发送与接收邮箱的)时间戳，分别存储在 CAN_RDTxR/CAN_TDTxR 寄存器中。内部定时器在每个 CAN 位时间(见 19.7.7 节)累加。内部定时器在接收和发送的帧起始位的采样点位置被采样，并生成时间戳。

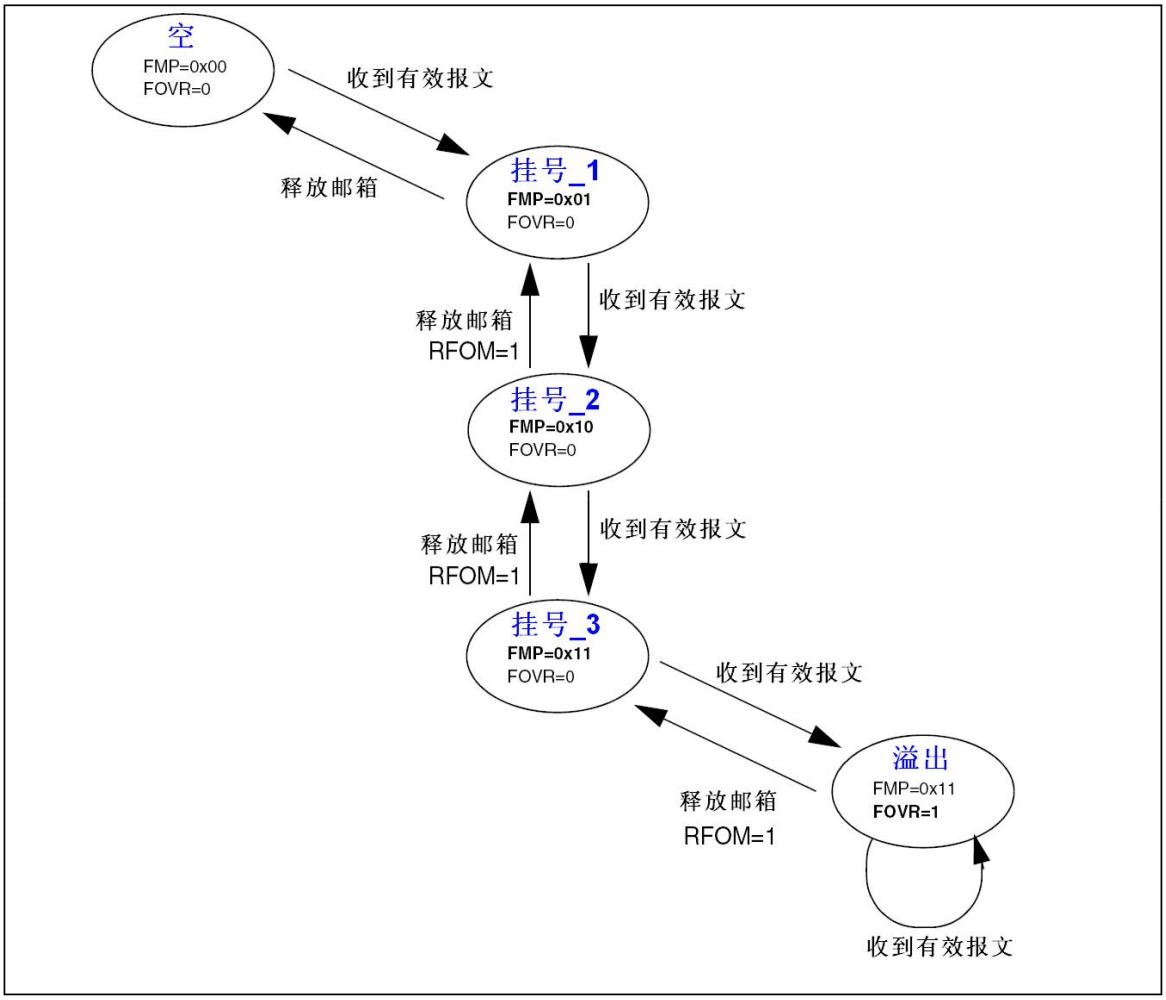
18.7.3 接收管理

接收到的报文，被存储在 3 级邮箱深度的 FIFO 中。FIFO 完全由硬件来管理，从而节省了 CPU 的处理负荷，简化了软件并保证了数据的一致性。应用程序只能通过读取 FIFO 输出邮箱，来读取FIFO 中最先收到的报文。

有效报文

根据 CAN 协议，当报文被正确接收(直到 EOF 域的最后一位都没有错误)，且通过了标识符过滤，那么该报文被认为是有效报文。请参考 19.7.4 节：标识符过滤。

图 146 接收 FIFO 状态



FIFO 管理

FIFO 从**空**状态开始，在接收到第一个有效的报文后，FIFO 状态变为**挂号_1**(pending_1)，硬件相应地把 CAN_RFR 寄存器的 FMP[1:0]设置为'01'(二进制 01b)。软件可以读取 FIFO 输出邮箱来读出邮箱中的报文，然后通过对 CAN_RFR 寄存器的 RFOM 位设置'1'来释放邮箱，这样 FIFO 又变为**空**状态了。如果在释放邮箱的同时，又收到了一个有效的报文，那么 FIFO 仍然保留在**挂号_1** 状态，软件可以读取 FIFO 输出邮箱来读出新收到的报文。

如果应用程序不释放邮箱，在接收到下一个有效的报文后，FIFO 状态变为**挂号_2**(pending_2)，硬件相应地把 FMP[1:0]设置为'10'(二进制 10b)。重复上面的过程，第三个有效的报文把 FIFO 变为**挂号_3** 状态(FMP[1:0]＝11b)。此时，软件必须对 RFOM 位设置 1 来释放邮箱，以便 FIFO 可以 有空间来存放下一个有效的报文；否则，下一个有效的报文到来时就会导致一个报文的丢失。参见 19.7.5 节：报文存储

溢出

当 FIFO 处于**挂号_3** 状态(即 FIFO 的 3 个邮箱都是满的)，下一个有效的报文就会导致**溢出**，并且一个报文会丢失。此时，硬件对 CAN_RFR 寄存器的 FOVR 位进行置'1'来表明溢出情况。至于哪个报文会被丢弃，取决于对 FIFO 的设置：

- 如果禁用了 FIFO 锁定功能(CAN_MCR 寄存器的 RFLM 位被清'0')，那么 FIFO 中最后收到的报文就被新报文所覆盖。这样，最新收到的报文不会被丢弃掉。
- 如果启用了 FIFO 锁定功能(CAN_MCR 寄存器的 RFLM 位被置'1')，那么新收到的报文就被

丢弃，软件可以读到 FIFO 中最早收到的 3 个报文。

接收相关的中断

一旦往 FIFO 存入一个报文，硬件就会更新 FMP[1:0]位，并且如果 CAN_IER 寄存器的 FMPIE 位为'1'，那么就会产生一个中断请求。

当FIFO 变满时(即第3个报文被存入)，CAN_RFR 寄存器的FULL 位就被置'1'，并且如果 CAN_IER 寄存器的 FFIE 位为'1'，那么就会产生一个满中断请求。

在溢出的情况下，FOVR 位被置'1'，并且如果 CAN_IER 寄存器的 FOVIE 位为'1'，那么就会产生一个溢出中断请求。

18.7.4 标识符过滤

在 CAN 协议里，报文的标识符不代表节点的地址，而是跟报文的内容相关的。因此，发送者乙广播的形式把报文发送给所有的接收者。节点在接收报文时－根据标识符的值－决定软件是否需要该报文；如果需要，就拷贝到 SRAM 里；如果不需要，报文就被丢弃且无需软件的干预。

为满足这一需求，bxCAN 控制器为应用程序提供了 14 个位宽可变的、可配置的过滤器组(13~0)，以便只接收那些软件需要的报文。硬件过滤的做法节省了 CPU 开销，否则就必须由软件过滤从而占用一定的 CPU 开销。每个过滤器组 x 由 2 个 32 位寄存器，CAN_FxR0 和 CAN_FxR1 组成。

可变的位宽

每个过滤器组的位宽都可以独立配置，以满足应用程序的不同需求。根据位宽的不同，每个过滤器组可提供：

- 1 个 32 位过滤器，包括：STDID[10:0]、EXTID[17:0]、IDE 和 RTR 位
- 2 个 16 位过滤器，包括：STDID[10:0]、IDE、RTR 和 EXTID[17:15]位

可参见图 147。

此外过滤器可配置为，屏蔽位模式和标识符列表模式。

屏蔽位模式

在屏蔽位模式下，标识符寄存器和屏蔽寄存器一起，指定报文标识符的任何一位，应该按照“必须匹配”或“不用关心”处理。

标识符列表模式

在标识符列表模式下，屏蔽寄存器也被当作标识符寄存器用。因此，不是采用一个标识符加一个屏蔽位的方式，而是使用 2 个标识符寄存器。接收报文标识符的每一位都必须跟过滤器标识符相同。

过滤器组位宽和模式的设置

过滤器组可以通过相应的CAN_FMR寄存器配置。在配置一个过滤器组前，必须通过清除 CAN_FAR 寄存器的 FACT 位，把它设置为禁用状态。通过设置 CAN_FS1R 的相应 FSCx 位，可以配置一个过滤器组的位宽，请参见图 147。通过 CAN_FMR 的 FBMx 位，可以配置对应的屏蔽/ 标识符寄存器的**标识符列表**模式或**屏蔽位**模式。

为了过滤出一组标识符，应该设置过滤器组工作在屏蔽位模式。

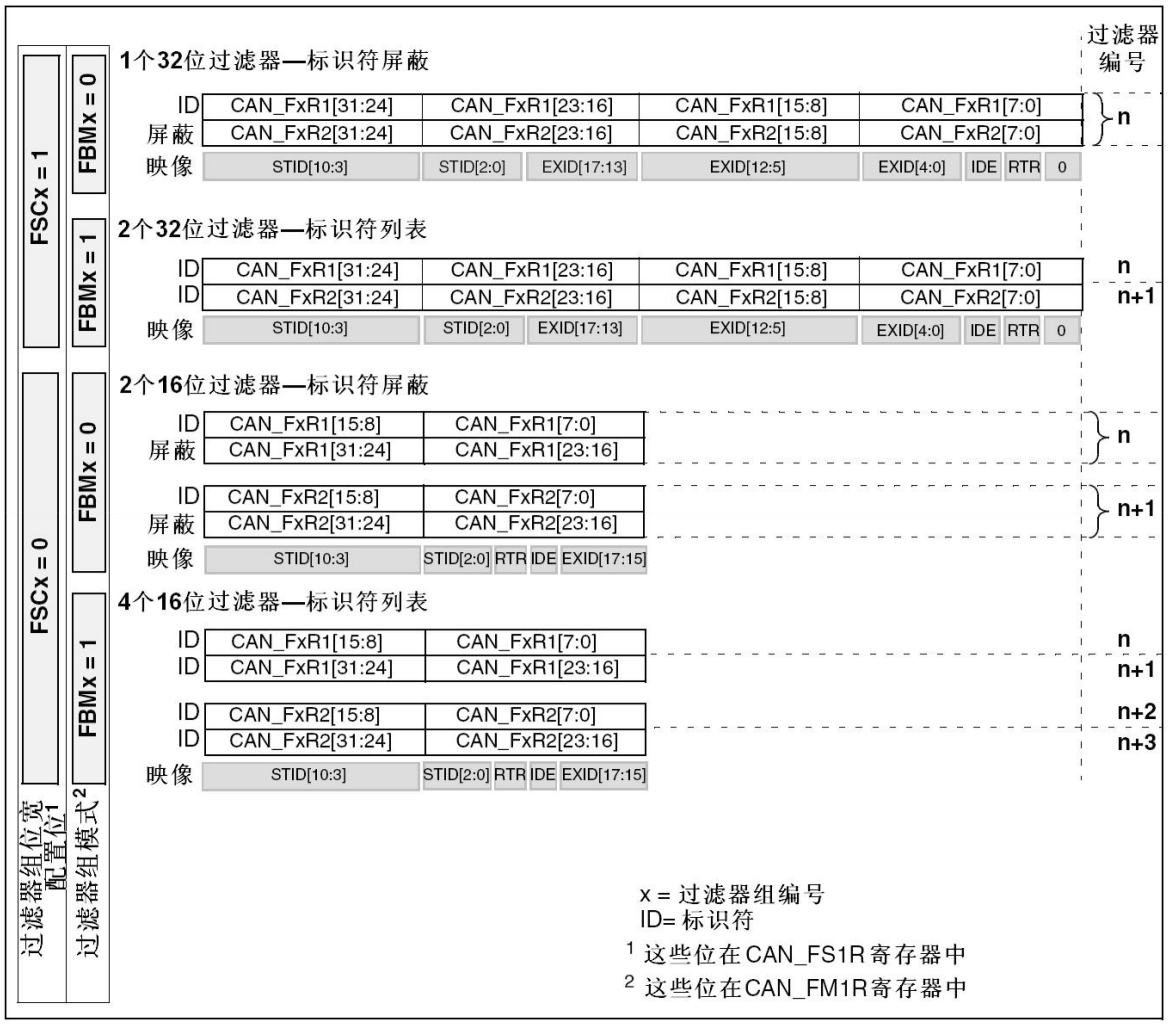
为了过滤出一个标识符，应该设置过滤器组工作在标识符列表模式。

应用程序不用的过滤器组，应该保持在禁用状态。

过滤器组中的每个过滤器，都被编号为(叫做过滤器号)从 0 开始，到某个最大数值－取决于过滤器组的模式和位宽的设置。

关于过滤器配置，参见下图。

图 147 过滤器组位宽设置－寄存器组织



过滤器匹配序号

一旦收到的报文被存入 FIFO，就可被应用程序访问。通常情况下，报文中的数据被拷贝到SRAM 中；为了把数据拷贝到合适的位置，应用程序需要根据报文的标识符来辨别不同的数据。bxCAN 提供了过滤器匹配序号，以简化这一辨别过程。

根据过滤器优先级规则，过滤器匹配序号和报文一起，被存入邮箱中。因此每个收到的报文，都有与它相关联的过滤器匹配序号。

过滤器匹配序号可以通过下面两种方式来使用：

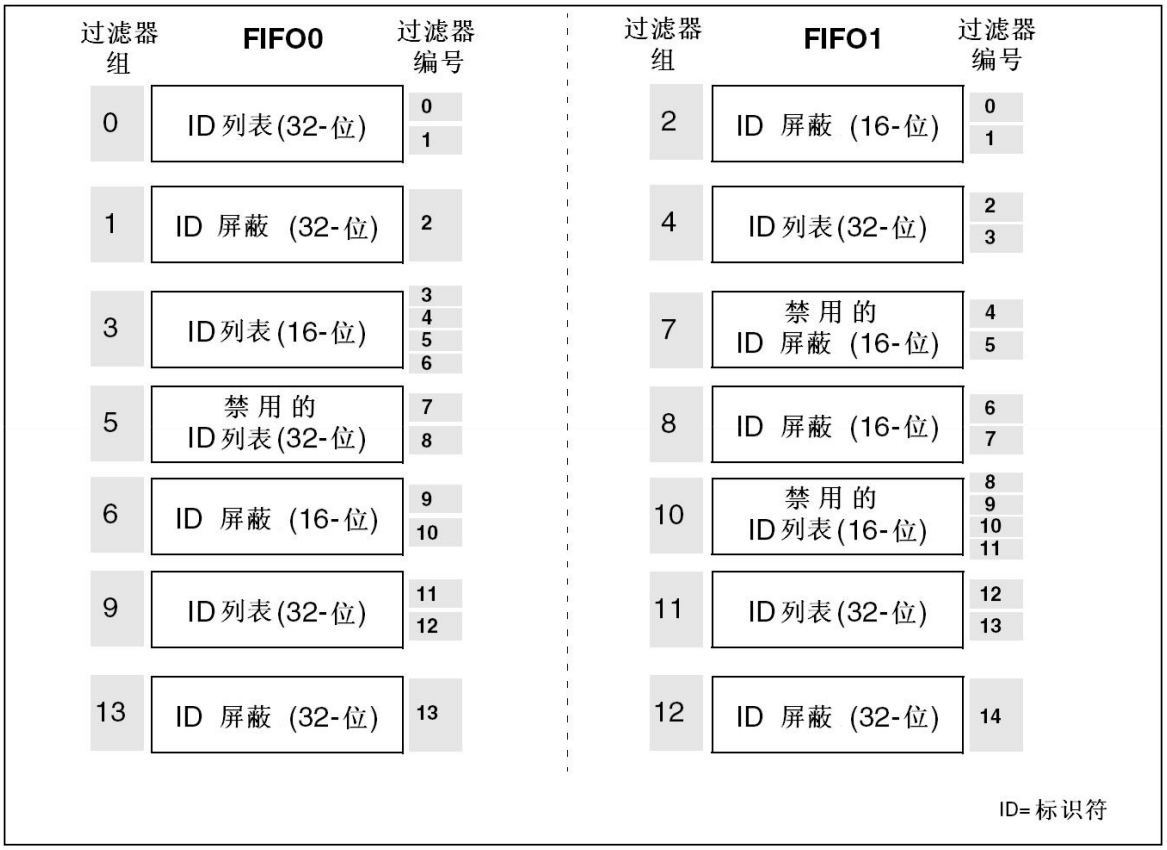
- 把过滤器匹配序号跟一系列所期望的值进行比较
- 把过滤器匹配序号当作一个索引来访问目标地址

对于标识符列表模式下的过滤器(非屏蔽方式的过滤器)，软件不需要直接跟标识符进行比较。

对于屏蔽位模式下的过滤器，软件只须对需要的那些屏蔽位(必须匹配的位)进行比较即可。

在给过滤器编号时，并不考虑过滤器组是否为激活状态。另外，每个 FIFO 各自对其关联的过滤器进行编号。请参考下图的例子。

图 148 过滤器编号的例子

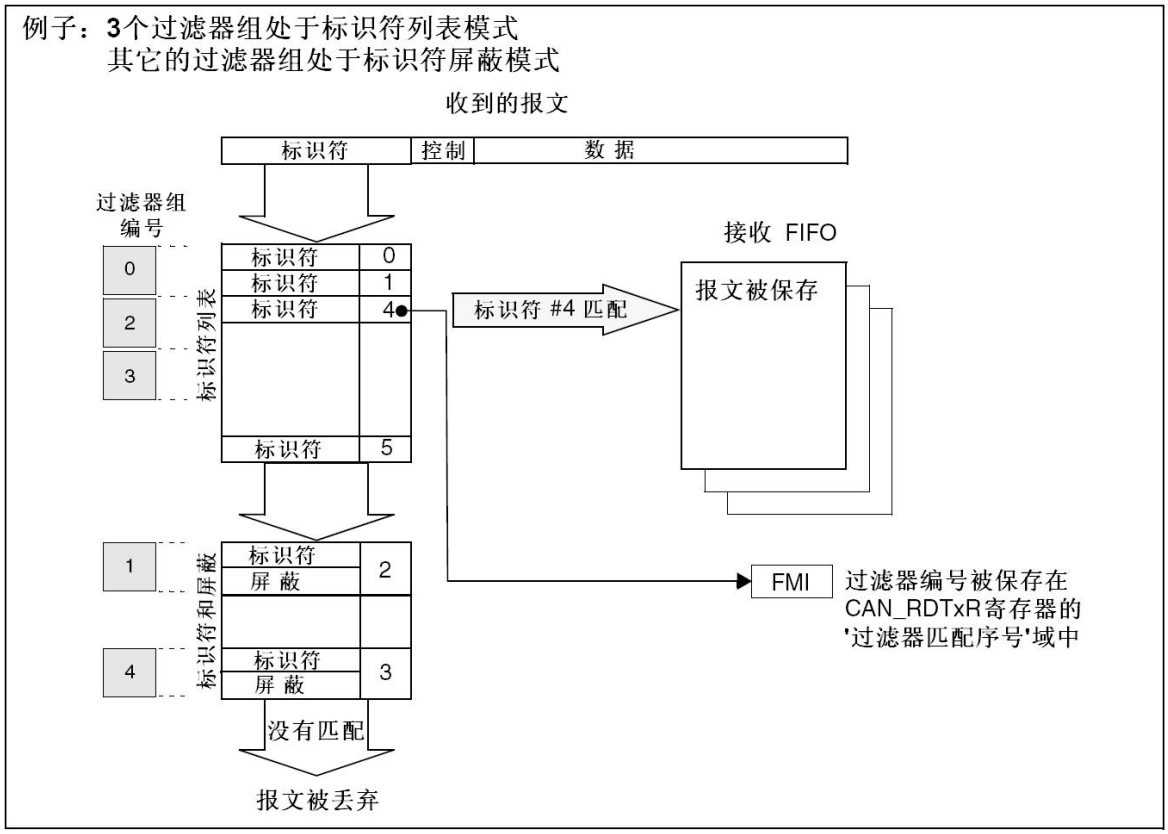


过滤器优先级规则

根据过滤器的不同配置，有可能一个报文标识符能通过多个过滤器的过滤；在这种情况下，存放在接收邮箱中的过滤器匹配序号，根据下列优先级规则来确定：

- 位宽为 32 位的过滤器，优先级高于位宽为 16 位的过滤器
- 对于位宽相同的过滤器，标识符列表模式的优先级高于屏蔽位模式
- 位宽和模式都相同的过滤器，优先级由过滤器号决定，过滤器号小的优先级高

图 149 过滤器机制的例子



上面的例子说明了 bxCAN 的过滤器规则：在接收一个报文时，其标识符首先与配置在标识符列表模式下的过滤器相比较；如果匹配上，报文就被存放到相关联的 FIFO 中，并且所匹配的过滤器的序号被存入过滤器匹配序号中。如同例子中所显示，报文标识符跟#4 标识符匹配，因此报 文内容和 FMI4 被存入 FIFO。

如果没有匹配，报文标识符接着与配置在屏蔽位模式下的过滤器进行比较。

如果报文标识符没有跟过滤器中的任何标识符相匹配，那么硬件就丢弃该报文，且不会对软件有任何打扰。

18.7.5 报文存储

邮箱是软件和硬件之间传递报文的接口。邮箱包含了所有跟报文有关的信息：标识符、数据、控制、状态和时间戳信息。

发送邮箱

软件需要在一个空的发送邮箱中，把待发送报文的各种信息设置好(然后再发出发送的请求)。发送的状态可通过查询 CAN_TSR 寄存器获知。

表 73 发送邮箱寄存器列表

相对发送邮箱基地址的偏移量	寄存器名
0	CAN_TIxR
4	CAN_TDTxR
8	CAN_TDLxR
12	CAN_TDHxR

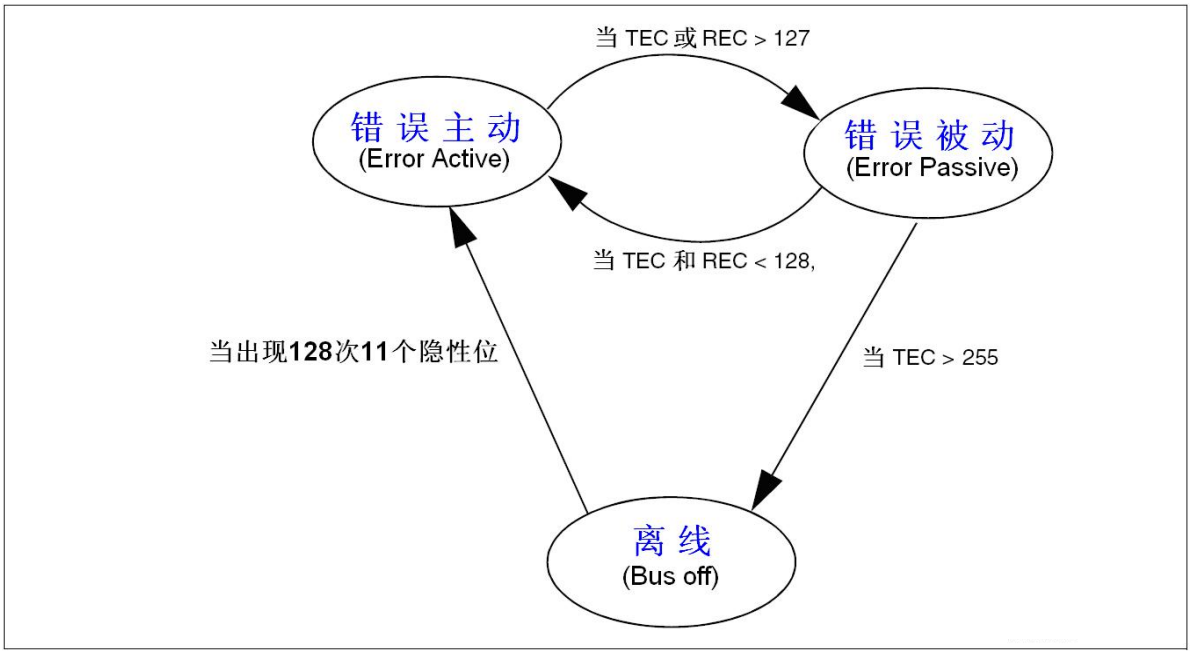
接收邮箱(FIFO)

在接收到一个报文后，软件就可以访问接收 FIFO 的输出邮箱来读取它。一旦软件处理了报文(如把它读出来)，软件就应该对 CAN_RFxR 寄存器的 RFOM 位进行置'1'，来释放该报文，以便为后面收到的报文留出存储空间。过滤器匹配序号存放在 CAN_RDTxR 寄存器的 FMI 域中。16 位的时间戳存放在 CAN_RDTxR 寄存器的 TIME[15:0]域中。

表 74 接收邮箱寄存器列表

相对接收邮箱基地址的偏移量	寄存器名
0	CAN_RIxR
4	CAN_RDTxR
8	CAN_RDLxR
12	CAN_RDHxR

图 150 CAN 错误状态图



18.7.6 出错管理

CAN 协议描述的出错管理，完全由硬件通过发送错误计数器(CAN_ESR 寄存器里的 TEC 域)，和接收错误计数器(CAN_ESR 寄存器里的 REC 域)来实现，其值根据错误的情况而增加或减少。关于 TEC 和 REC 管理的详细信息，请参考 CAN 标准。

软件可以读出它们的值来判断 CAN 网络的稳定性。

此外，CAN_ESR 寄存器提供了当前错误状态的详细信息。通过设置 CAN_IER 寄存器(比如 ERRIE 位)，当检测到出错时软件可以灵活地控制中断的产生。

离线恢复

当 TEC 大于 255 时，bxCAN 就进入离线状态，同时 CAN_ESR 寄存器的 BOFF 位被置'1'。在离线状态下，bxCAN 无法接收和发送报文。

根据 CAN_MCR 寄存器中 ABOM 位的设置，bxCAN 可以自动或在软件的请求下，从离线状态恢复(变为错误主动状态)。在这两种情况下，bxCAN 都必须等待一个 CAN 标准所描述的恢复过程(CAN RX 引脚上检测到 128 次 11 个连续的隐性位)。

如果 ABOM 位为'1'，bxCAN 进入离线状态后，就自动开启恢复过程。

如果 ABOM 位为'0'，软件必须先请求 bxCAN 进入然后再退出初始化模式，随后恢复过程才被开启。

注：　在初始化模式下，bxCAN 不会监视 CAN RX 引脚的状态，这样就不能完成恢复过程。
为了完成恢复过程，bxCAN 必须工作在正常模式。

18.7.7 位时间特性

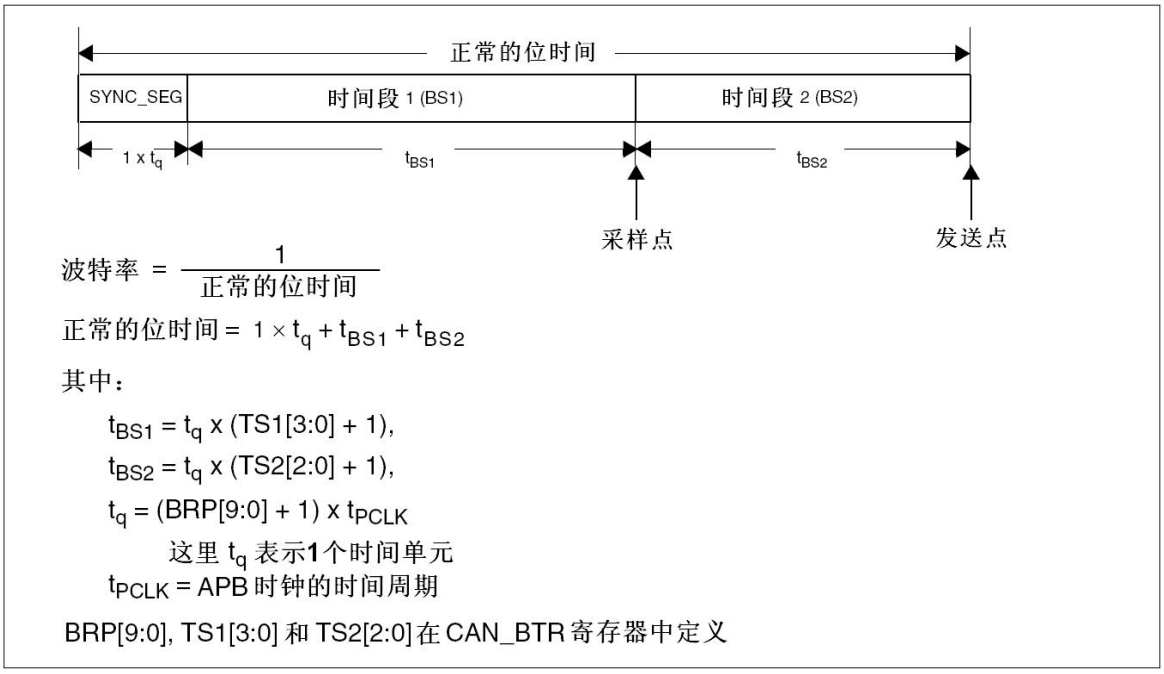
位时间特性逻辑通过采样来监视串行的 CAN 总线，并且通过与帧起始位的边沿进行同步，及通 过与后面的边沿进行重新同步，来调整其采样点。
它的操作可以简单解释为，如下所述把名义上的每位时间分为 3 段：

- **同步段(SYNC_SEG)**：通常期望位的变化发生在该时间段内。其值固定为 1 个时间单元(1 x t_{CAN})。
- **时间段 1(BS1)**：定义采样点的位置。它包含 CAN 标准里的 PROP_SEG 和 PHASE_SEG1。其值可以编程为 1 到 16 个时间单元，但也可以被自动延长，以补偿因为网络中不同节点的频率差异所造成的相位的正向漂移。
- **时间段 2(BS2)**：定义发送点的位置。它代表 CAN 标准里的 PHASE_SEG2。其值可以编程为 1 到 8 个时间单元，但也可以被自动缩短以补偿相位的负向漂移。

重新同步跳跃宽度(SJW)定义了，在每位中可以延长或缩短多少个时间单元的上限。其值可以编程为 1 到 4 个时间单元。
有效跳变被定义为，当 bxCAN 自己没有发送隐性位时，从显性位到隐性位的第 1 次转变。
如果在时间段 1(BS1)而不是在同步段(SYNC_SEG)检测到有效跳变，那么 BS1 的时间就被延长最多 SJW 那么长，从而采样点被延迟了。
相反如果在时间段 2(BS2)而不是在 SYNC_SEG 检测到有效跳变，那么 BS2 的时间就被缩短最多 SJW 那么长，从而采样点被提前了。
为了避免软件的编程错误，对位时间特性寄存器(CAN_BTR)的设置，只能在 bxCAN 处于初始化状态下进行。

注：　关于 CAN 位时间特性和重同步机制的详细信息，请参考 ISO11898 标准。

图 151 位时序



$$波特率 = \frac{1}{正常的位时间}$$

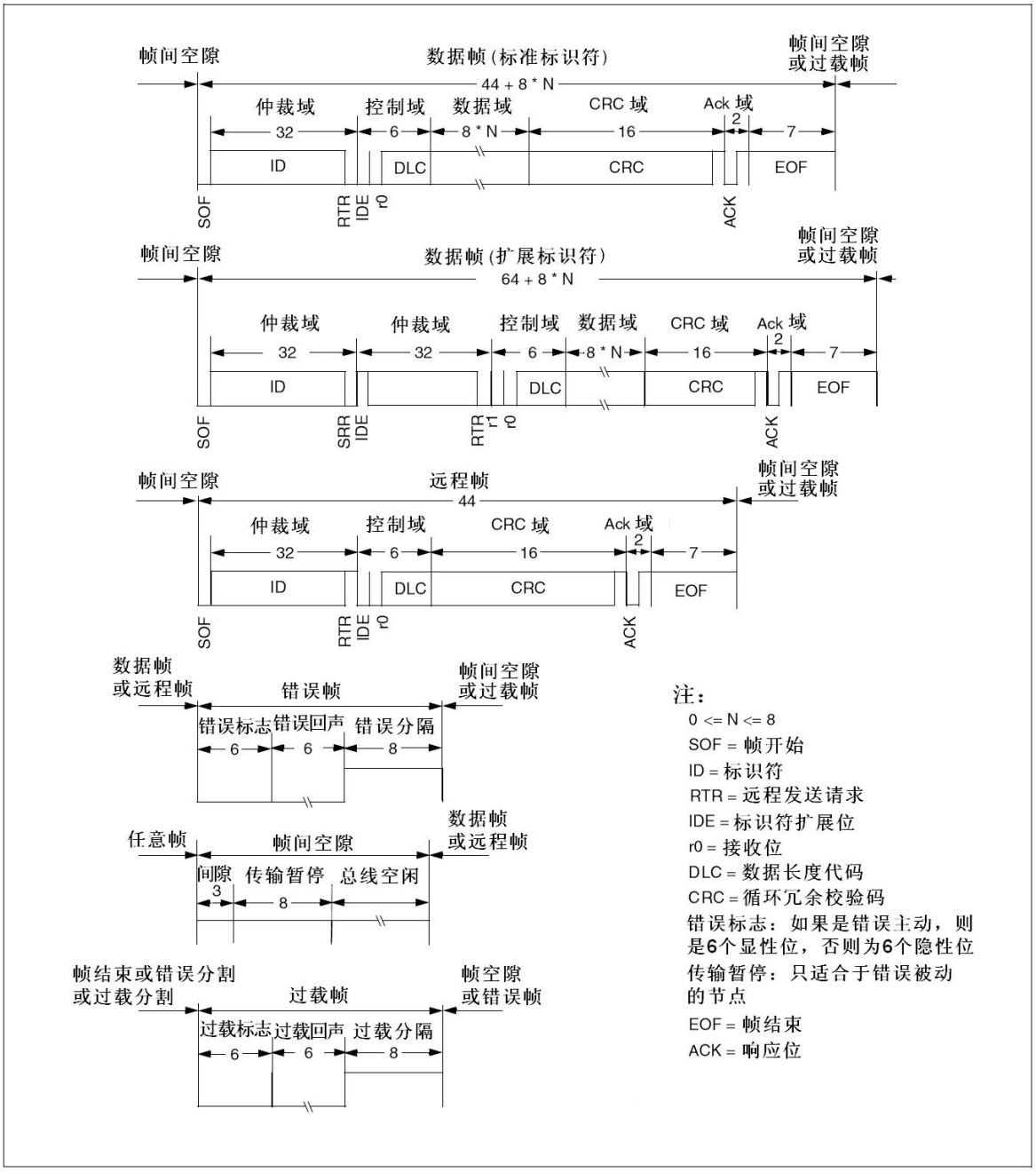
$$正常的位时间 = 1 \times t_q + t_{BS1} + t_{BS2}$$

其中：

$t_{BS1} = t_q \times (TS1[3:0] + 1)$,
$t_{BS2} = t_q \times (TS2[2:0] + 1)$,
$t_q = (BRP[9:0] + 1) \times t_{PCLK}$
这里 t_q 表示1个时间单元
t_{PCLK} = APB 时钟的时间周期

BRP[9:0], TS1[3:0] 和 TS2[2:0]在 CAN_BTR 寄存器中定义

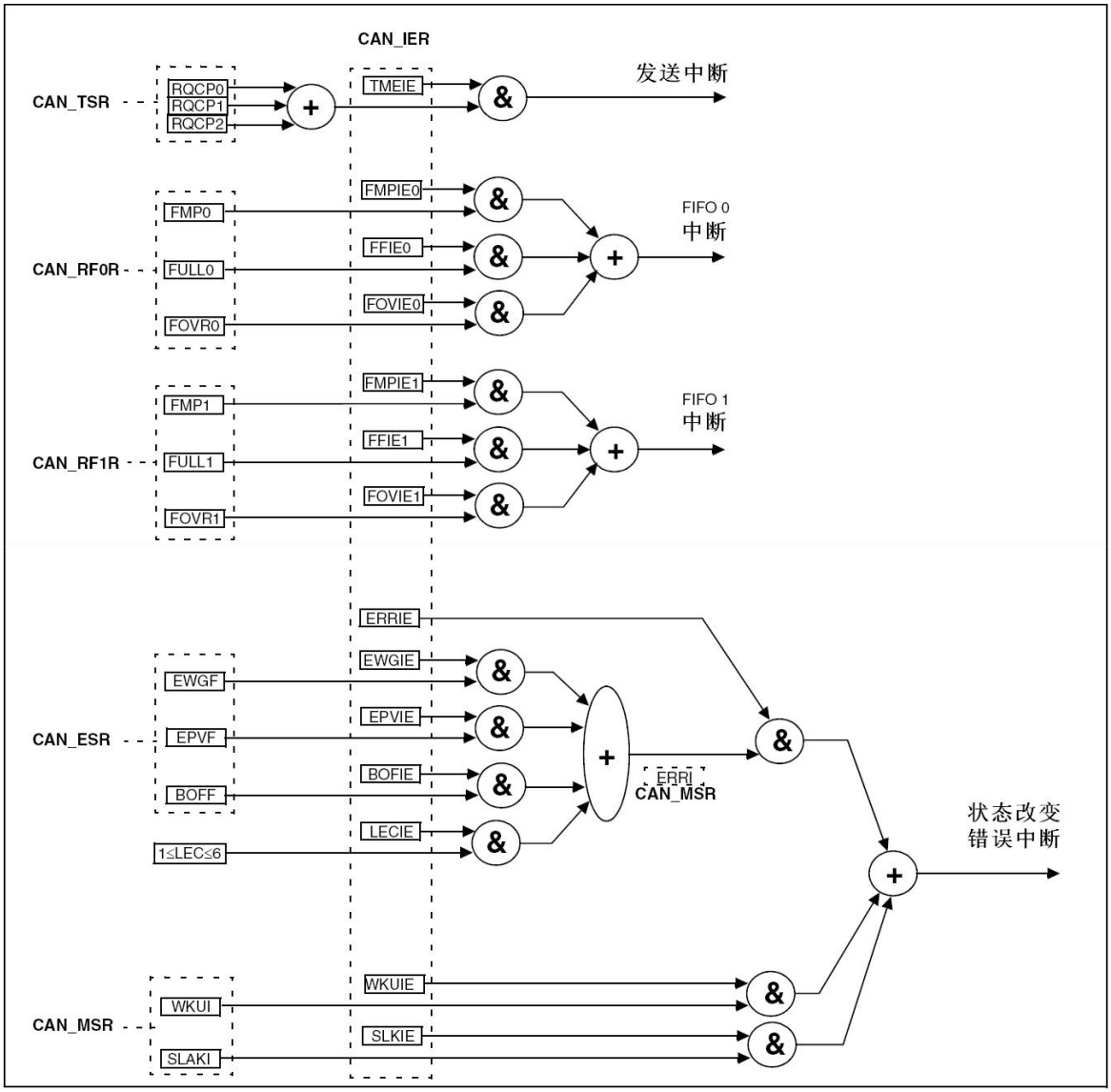
图 152 各种 CAN 帧


帧间空隙
数据帧(标准标识符)
帧间空隙或过载帧
44 + 8 * N
仲裁域
控制域
数据域
CRC 域
Ack 域
32
6
8 * N
16
2
7
ID
DLC
CRC
EOF
SOF
RTR
IDE
r0
ACK
帧间空隙
数据帧(扩展标识符)
帧间空隙或过载帧
64 + 8 * N
仲裁域
仲裁域
控制域
数据域
CRC 域
Ack 域
32
32
6
8 * N
16
2
7
ID
DLC
CRC
EOF
SOF
SRR
IDE
RTR
r1
r0
ACK
帧间空隙
远程帧
帧间空隙或过载帧
44
仲裁域
控制域
CRC 域
Ack 域
32
6
16
2
7
ID
DLC
CRC
EOF
SOF
RTR
IDE
r0
ACK
数据帧或远程帧
错误帧
帧间空隙或过载帧
错误标志
错误回声
错误分隔
6
6
8
任意帧
帧间空隙
数据帧或远程帧
间隙
传输暂停
总线空闲
3
8
帧结束或错误分割或过载分割
过载帧
帧空隙或错误帧
过载标志
过载回声
过载分隔
6
6
8
注：
0 <= N <= 8
SOF = 帧开始
ID = 标识符
RTR = 远程发送请求
IDE = 标识符扩展位
r0 = 接收位
DLC = 数据长度代码
CRC = 循环冗余校验码
错误标志：如果是错误主动，则是6个显性位，否则为6个隐性位
传输暂停：只适合于错误被动的节点
EOF = 帧结束
ACK = 响应位

18.8 bxCAN 中断

bxCAN 占用 4 个专用的中断向量。通过设置 CAN 中断允许寄存器(CAN_IER)，每个中断源都可以单独允许和禁用。

图 153 事件标志和中断产生



- 发送中断可由下列事件产生：
  - 发送邮箱 0 变为空，CAN_TSR 寄存器的 RQCP0 位被置'1'。
  - 发送邮箱 1 变为空，CAN_TSR 寄存器的 RQCP1 位被置'1'。
  - 发送邮箱 2 变为空，CAN_TSR 寄存器的 RQCP2 位被置'1'。
- FIFO0 中断可由下列事件产生：
  - FIFO0 接收到一个新报文，CAN_RF0R 寄存器的 FMP0 位不再是'00'。
  - FIFO0 变为满的情况，CAN_RF0R 寄存器的 FULL0 位被置'1'。
  - FIFO0 发生溢出的情况，CAN_RF0R 寄存器的 FOVR0 位被置'1'。
- FIFO1 中断可由下列事件产生：
  - FIFO1 接收到一个新报文，CAN_RF1R 寄存器的 FMP1 位不再是'00'。
  - FIFO1 变为满的情况，CAN_RF1R 寄存器的 FULL1 位被置'1'。
  - FIFO1 发生溢出的情况，CAN_RF1R 寄存器的 FOVR1 位被置'1'。
- **错误和状态变化中断**可由下列事件产生：

- 出错情况，关于出错情况的详细信息请参考 CAN 错误状态寄存器(CAN_ESR)。
- 唤醒情况，在 CAN 接收引脚上监视到帧起始位(SOF)。
- CAN 进入睡眠模式。

18.9 CAN 寄存器描述

关于寄存器描述中所用到的缩略词可参见第 2.1 节。必须以字(32 位)的方式操作这些外设寄存器。

18.9.1 寄存器访问保护

对某些寄存器的错误访问会导致一个 CAN 节点对整个 CAN 网络的暂时性干扰。因此，软件只能在 CAN 处于初始化模式时修改 CAN_BTR 寄存器。

虽然错误数据的发送对 CAN 网的网络层不会带来问题，但却会对应用程序造成严重影响。因此，软件只能在发送邮箱为空的状态改变它，请参见图 145。

过滤器的数值只能在关闭对应过滤器组的状态下，或设置 FINIT 位为'1'后才能修改。此外，只有在设置整个过滤器为初始化模式下(即 FINIT=1)，才能修改过滤器的设置，即修改 CAN_FMxR，CAN_FSxR 和 CAN_FFAR 寄存器。

18.9.2 CAN 控制和状态寄存器

有关寄存器说明中的缩写，参见 2.1 节。

CAN 主控制寄存器 (CAN_MCR)

地址偏移量: 0x00

复位值: 0x0001 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															DBF
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET	保留							TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
rs	res							rw	rw	rw	rw	rw	rw	rw	rw

位 31:15	保留，硬件强制为 0。
位 16	**DBF：**调试冻结(Debug freeze) 0：在调试时，CAN 照常工作 1：在调试时，冻结 CAN 的接收/发送。仍然可以正常地读写和控制接收 FIFO。
位 15	**RESET:** bxCAN 软件复位(bxCAN software master reset) 0：本外设正常工作； 1：对 bxCAN 进行强行复位，复位后 bxCAN 进入睡眠模式(FMP 位和 CAN_MCR 寄存器被初始化为其复位值)。此后硬件自动对该位清'0'。
位 14:8	保留，硬件强制为 0。
位 7	**TTCM:** 时间触发通信模式(Time triggered communication mode) 0：禁止时间触发通信模式； 1：允许时间触发通信模式。 注: 要想了解关于时间触发通信模式的更多信息，请参考 19.7.2：时间触发通信模式。
位 6	**ABOM:** 自动离线(Bus-Off)管理(Automatic bus-off management) 该位决定 CAN 硬件在什么条件下可以退出离线状态。 0：离线状态的退出过程是，软件对 CAN_MCR 寄存器的 INRQ 位进行置'1'随后清'0'后，一旦硬件检测到 128 次 11 位连续的隐性位，则退出离线状态；

	1：一旦硬件检测到 128 次 11 位连续的隐性位，则自动退出离线状态。 注：关于离线状态的更多信息，请参考 19.7.6：出错管理。

位 5	**AWUM:** 自动唤醒模式(Automatic wakeup mode) 该位决定 CAN 处在睡眠模式时由硬件还是软件唤醒 0：睡眠模式通过清除 CAN_MCR 寄存器的 SLEEP 位，由软件唤醒； 1：睡眠模式通过检测 CAN 报文，由硬件自动唤醒。唤醒的同时，硬件自动对 CAN_MSR 寄存器的 SLEEP 和 SLAK 位清'0' 。
位 4	**NART:** 禁止报文自动重传(No automatic retransmission) 0：按照 CAN 标准，CAN 硬件在发送报文失败时会一直自动重传直到发送成功； 1：CAN 报文只被发送 1 次，不管发送的结果如何(成功、出错或仲裁丢失)。
位 3	**RFLM:** 接收 FIFO 锁定模式(Receive FIFO locked mode) 0：在接收溢出时 FIFO 未被锁定，当接收 FIFO 的报文未被读出，下一个收到的报文会覆盖原有的报文； 1：在接收溢出时 FIFO 被锁定，当接收 FIFO 的报文未被读出，下一个收到的报文会被丢弃。
位 2	**TXFP:** 发送 FIFO 优先级(Transmit FIFO priority) 当有多个报文同时在等待发送时，该位决定这些报文的发送顺序 0：优先级由报文的标识符来决定； 1：优先级由发送请求的顺序来决定。
位 1	**SLEEP:** 睡眠模式请求(Sleep mode request) 软件对该位置'1'可以请求 CAN 进入睡眠模式，一旦当前的 CAN 活动(发送或接收报文)结束，CAN 就进入睡眠。 软件对该位清'0'使 CAN 退出睡眠模式。 当设置了 AWUM 位且在 CAN Rx 信号中检测出 SOF 位时，硬件对该位清'0'。 在复位后该位被置'1'，即 CAN 在复位后处于睡眠模式。
位 0	**INRQ**: 初始化请求(Initialization request) 软件对该位清'0'可使 CAN 从初始化模式进入正常工作模式：当 CAN 在接收引脚检测到连续的 11 个隐性位后，CAN 就达到同步，并为接收和发送数据作好准备了。为此，硬件相应 CAN_MSR 寄存器的 INAK 位清'0'。 软件对该位置 1 可使 CAN 从正常工作模式进入初始化模式：一旦当前的 CAN 活动(发送或接收)束，CAN 就进入初始化模式。相应地，硬件对 CAN_MSR 寄存器的 INAK 位置'1'。

CAN 主状态寄存器(CAN_MSR)

地址偏移量: 0x04

复位值: 0x0000 0C02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				RX	SAMP	RXM	TXM	保留			SLAKI	WKUI	ERRI	SLAK	INAK
				r	r	r	r				rc_w1	rc_w1	rc_w1	r	r

位 31:12	保留位，硬件强制为 0
位 11	**RX**：CAN 接收电平(CAN Rx signal) 该位反映 CAN 接收引脚(CAN_RX)的实际电平。
位 10	**SAMP**：上次采样值(Last sample point) CAN 接收引脚的上次采样值(对应于当前接收位的值)。
位 9	**RXM**：接收模式(Receive mode) 该位为'1'表示 CAN 当前为接收器。
位 8	**TXM**：发送模式(Transmit mode) 该位为'1'表示 CAN 当前为发送器。

位 7:5	保留位，硬件强制为 0。

位 4	**SLAKI**：睡眠确认中断(Sleep acknowledge interrupt) 当 SLKIE=1，一旦 CAN 进入睡眠模式硬件就对该位置’1，紧接着相应的中断被触发。当设置该位为’1’时，如果设置了 CAN_IER 寄存器中的 SLKIE 位，将产生一个状态改变中断。 软件可对该位清’0’，当 SLAK 位被清’0’时硬件也对该位清’0’。 注: 当 SLKIE=0, 不应该查询该位，而应该查询 SLAK 位来获知睡眠状态。
位 3	**WKUI**：唤醒中断挂号(Wakeup interrupt) 当 CAN 处于睡眠状态，一旦检测到帧起始位(SOF)，硬件就置该位为’1’；并且如果 CAN_IER 寄存器的 WKUIE 位为’1’，则产生一个状态改变中断。 该位由软件清’0’。
位 2	**ERRI**：出错中断挂号(Error interrupt) 当检测到错误时，CAN_ESR 寄存器的某位被置’1’，如果 CAN_IER 寄存器的相应中断使能位也被置’1’时，则硬件对该位置’1’；如果 CAN_IER 寄存器的 ERRIE 位为’1’，则产生状态改变中断。 该位由软件清’0’。
位 1	**SLAK**：睡眠模式确认 该位由硬件置’1’，指示软件 CAN 模块正处于睡眠模式。该位是对软件请求进入睡眠模式的确认(对 CAN_MCR 寄存器的 SLEEP 位置’1’)。 当 CAN 退出睡眠模式时硬件对该位清’0’ (需要跟 CAN 总线同步)。这里跟 CAN 总线同步是指，硬 件需要在 CAN 的 RX 引脚上检测到连续的 11 位隐性位。 注: 通过软件或硬件对 CAN_MCR 的 SLEEP 位清’0’，将启动退出睡眠模式的过程。有关清除 SLEEP 位的详细信息，参见 CAN_MCR 寄存器的 AWUM 位的描述。
位 0	**INAK**：初始化确认 该位由硬件置’1’，指示软件 CAN 模块正处于初始化模式。该位是对软件请求进入初始化模式的确认(对 CAN_MCR 寄存器的 INRQ 位置’1’)。 当 CAN 退出初始化模式时硬件对该位清’0’ (需要跟 CAN 总线同步)。这里跟 CAN 总线同步是指，硬件需要在 CAN 的 RX 引脚上检测到连续的 11 位隐性位。

CAN 发送状态寄存器(CAN_TSR)

地址偏移量: 0x08

复位值: 0x1C00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOW2	LOW1	LOW0	TME2	TME1	TME0	CODE[1:0]		ABRQ2	保留			TERR2	ALST2	TXOK2	RQCP2
r	r	r	r	r	r	r	r	rs	res			rc w1	rc w1	rc w1	rc w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRQ1	保留			TERR1	ALST1	TXOK1	RQCP1	ABRQ0	保留			TERR0	ALST0	TXOK0	RQCP0
rs	res			rc w1	rc w1	rc w1	rc w1	rs	res			rc w1	rc w1	rc w1	rc w1

位 31	**LOW2:** 邮箱 2 最低优先级标志(Lowest priority flag for mailbox 2) 当多个邮箱在等待发送报文，且邮箱 2 的优先级最低时，硬件对该位置'1'。
位 30	**LOW1:** 邮箱 1 最低优先级标志(Lowest priority flag for mailbox 1) 当多个邮箱在等待发送报文，且邮箱 1 的优先级最低时，硬件对该位置'1'。
位 29	**LOW0:** 邮箱 0 最低优先级标志(Lowest priority flag for mailbox 0) 当多个邮箱在等待发送报文，且邮箱 0 的优先级最低时，硬件对该位置'1'。 注：如果只有 1 个邮箱在等待，则 LOW[2:0]被清'0'。
位 28	**TME2:** 发送邮箱 2 空(Transmit mailbox 2 empty) 当邮箱 2 中没有等待发送的报文时，硬件对该位置'1'。
位 27	**TME1:** 发送邮箱 1 空(Transmit mailbox 1 empty) 当邮箱 1 中没有等待发送的报文时，硬件对该位置'1'。
位 26	**TME0:** 发送邮箱 0 空(Transmit mailbox 0 empty) 当邮箱 0 中没有等待发送的报文时，硬件对该位置'1'。

位 25:24	**CODE[1:0]:** 邮箱号(Mailbox code) 当有至少 1 个发送邮箱为空时，这 2 位表示下一个空的发送邮箱号。 当所有的发送邮箱都为空时，这 2 位表示优先级最低的那个发送邮箱号。
位 23	**ABRQ2:** 邮箱 2 中止发送(Abort request for mailbox 2) 软件对该位置'1'，可以中止邮箱 2 的发送请求，当邮箱 2 的发送报文被清除时硬件对该位清'0'。 如果邮箱 2 中没有等待发送的报文，则对该位置'1'没有任何效果。
位 22:20	保留位，硬件强制其值为 0
位 19	**TERR2:** 邮箱 2 发送失败(Transmission error of mailbox 2) 当邮箱 2 因为出错而导致发送失败时，对该位置'1'。
位 18	**ALST2:** 邮箱 2 仲裁丢失(Arbitration lost for mailbox 2) 当邮箱 2 因为仲裁丢失而导致发送失败时，对该位置'1'。
位 17	**TXOK2**: 邮箱 2 发送成功 (Transmission OK of mailbox 2) 每次在邮箱 2 进行发送尝试后，硬件对该位进行更新： 0：上次发送尝试失败； 1：上次发送尝试成功。 当邮箱 2 的发送请求被成功完成后，硬件对该位置'1'。请参见图 145。
位 16	**RQCP2:** 邮箱 2 请求完成(Request completed mailbox 2) 当上次对邮箱 2 的请求(发送或中止)完成后，硬件对该位置'1'。 软件对该位写'1'可以对其清'0'；当硬件接收到发送请求时也对该位清'0'(CAN_TI2R 寄存器的 TXRQ 位被置'1')。 该位被清'0'时，邮箱 2 的其它发送状态位(TXOK2, ALST2 和 TERR2)也被清'0'。
位 15	**ABRQ1:** 邮箱 1 中止发送(Abort request for mailbox 1) 软件对该位置'1'，可以中止邮箱 1 的发送请求，当邮箱 1 的发送报文被清除时硬件对该位清'0'。 如果邮箱 1 中没有等待发送的报文，则对该位置'1'没有任何效果。
位 14:12	保留位，硬件强制其值为 0
位 11	**TERR1:** 邮箱 1 发送失败(Transmission error of mailbox 1) 当邮箱 1 因为出错而导致发送失败时，对该位置'1'。
位 10	**ALST1:** 邮箱 1 仲裁丢失(Arbitration lost for mailbox 1) 当邮箱 1 因为仲裁丢失而导致发送失败时，对该位置'1'。
位 9	**TXOK1**: 邮箱 1 发送成功 (Transmission OK of mailbox 1) 每次在邮箱 1 进行发送尝试后，硬件对该位进行更新：0： 上次发送尝试失败； 1：上次发送尝试成功。 当邮箱 1 的发送请求被成功完成后，硬件对该位置'1'。请参见图 145。

位 8	**RQCP1:** 邮箱 1 请求完成(Request completed mailbox 1) 当上次对邮箱 1 的请求(发送或中止)完成后，硬件对该位置'1'。 软件对该位写'1'可以对其清'0'；当硬件接收到发送请求时也对该位清'0'(CAN_TI1R 寄存器的 TXRQ 位被置'1')。 该位被清'0'时，邮箱 1 的其它发送状态位(TXOK1, ALST1 和 TERR1)也被清'0'。
位 7	**ABRQ0:** 邮箱 0 中止发送(Abort request for mailbox 0) 软件对该位置'1'可以中止邮箱 0 的发送请求，当邮箱 0 的发送报文被清除时硬件对该位清'0'。 如果邮箱 0 中没有等待发送的报文，则对该位置 1 没有任何效果。
位 6:4	保留位，硬件强制其值为 0
位 3	**TERR0:** 邮箱 0 发送失败(Transmission error of mailbox 0) 当邮箱 0 因为出错而导致发送失败时，对该位置'1'。
位 2	**ALST0:** 邮箱 0 仲裁丢失(Arbitration lost for mailbox 0) 当邮箱 0 因为仲裁丢失而导致发送失败时，对该位置'1'。
位 1	**TXOK0**: 邮箱 0 发送成功 (Transmission OK of mailbox 0) 每次在邮箱 0 进行发送尝试后，硬件对该位进行更新： 0：上次发送尝试失败； 1：上次发送尝试成功。 当邮箱 0 的发送请求被成功完成后，硬件对该位置'1'。请参见图 145。
位 0	**RQCP1:** 邮箱 0 请求完成(Request completed mailbox 0) 当上次对邮箱 0 的请求(发送或中止)完成后，硬件对该位置'1'。 软件对该位写'1'可以对其清'0'；当硬件接收到发送请求时也对该位清'0'(CAN_TI0R 寄存器的 TXRQ 位被置'1')。 该位被清'0'时，邮箱 0 的其它发送状态位(TXOK0, ALST0 和 TERR0)也被清'0'。

CAN 接收 FIFO 0 寄存器(CAN_RF0R)

地址偏移量: 0x0C

复位值: 0x00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
res															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										RFOM0	FOVR0	FULL0	保留	FMP0[1:0]	
res										rs	rc_wl	rc_wl		r	r

位 31:6	保留位，硬件强制为 0
位 5	**RFOM0:** 释放接收 FIFO 0 输出邮箱(Release FIFO 0 output mailbox) 软件通过对该位置'1'来释放接收 FIFO 的输出邮箱。如果接收 FIFO 为空，那么对该位置'1'没有任何效果，即只有当 FIFO 中有报文时对该位置'1'才有意义。如果 FIFO 中有 2 个以上的报文，由于 FIFO 的特点，软件需要释放输出邮箱才能访问第 2 个报文。 当输出邮箱被释放时，硬件对该位清'0'。
位 4	**FOVR0:** FIFO 0 溢出(FIFO 0 overrun) 当 FIFO 0 已满，又收到新的报文且报文符合过滤条件，硬件对该位置'1'。 该位由软件清'0'。
位 3	**FULL0:** FIFO 0 满(FIFO 0 full) 当 FIFO 0 中有 3 个报文时，硬件对该位置'1'。 该位由软件清'0'。
位 2	保留位，硬件强制其值为 0

位 1:0	**FMP0[1:0]:** FIFO 0 报文数目(FIFO 0 message pending) FIFO 0 报文数目这 2 位反映了当前接收 FIFO 0 中存放的报文数目。 每当 1 个新的报文被存入接收 FIFO 0，硬件就对 FMP0 加 1。 每当软件对 RFOM0 位写'1'来释放输出邮箱，FMP0 就被减 1，直到其为 0。

CAN 接收 FIFO 1 寄存器(CAN_RF1R)

地址偏移量: 0x10

复位值: 0x00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										RFOM1	FOVR1	FULL1	保留	FMP1[1:0]	
										rs	rc wl	rc wl		r	r

位 31：6	保留，硬件强制为 0
位 5	**RFOM1:** 释放接收 FIFO 1 输出邮箱(Release FIFO 1 output mailbox) 软件通过对该位置'1'来释放接收 FIFO 的输出邮箱。如果接收 FIFO 为空，那么对该位置'1'没有任何效果，即只有当 FIFO 中有报文时对该位置'1'才有意义。如果 FIFO 中有 2 个以上的报文，由于 FIFO 的特点，软件需要释放输出邮箱才能访问第 2 个报文。 当输出邮箱被释放时，硬件对该位清'0'。
位 4	**FOVR1:** FIFO 1 溢出(FIFO 1 overrun) 当 FIFO 1 已满，又收到新的报文且报文符合过滤条件，硬件对该位置'1'。 该位由软件清'0'。
位 3	**FULL1:** FIFO 1 满(FIFO 1 full) 当 FIFO 1 中有 3 个报文时，硬件对该位置' 1' 。 该位由软件清'0'。
位 2	保留位，硬件强制其值为 0
位 1:0	**FMP1[1:0]:** FIFO 1 报文数目(FIFO 1 message pending) FIFO 1 报文数目这 2 位反映了当前接收 FIFO 1 中存放的报文数目。 每当 1 个新的报文被存入接收 FIFO 1，硬件就对 FMP1 加 1。 每当软件对 RFOM1 位写 1 来释放输出邮箱，FMP1 就被减 1，直到其为 0。

CAN 中断使能寄存器(CAN_IER)

地址偏移量: 0x14

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留														SLKIE	WKUIE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIE	保留			LECIE	BOFIE	EPVIE	EWGIE	保留	FOVIE1	FFIE1	FMPIE1	FOVIE0	FFIE0	FMPIE0	TMEIE
rw	res			rw	rw	rw	rw	res	rw	rw	rw	rw	rw	rw	rw

位 31:18	保留位，硬件强制为 0
位 17	**SLKIE**：睡眠中断使能(Sleep interrupt enable) 0：当 SLAKI 位被置'1'时，不产生中断； 1：当 SLAKI 位被置'1'时，产生中断。
位 16	**WKUIE:** 唤醒中断使能(Wakeup interrupt enable) 0：当 WKUI 位被置'1'时，不产生中断； 1：当 WKUI 位被置'1'时，产生中断。
位 15	**ERRIE:** 错误中断使能(Error interrupt enable) 0：当 CAN_ESR 寄存器有错误挂号时，不产生中断； 1：当 CAN_ESR 寄存器有错误挂号时，产生中断。
位 14:12	保留位，硬件强制为 0。
位 11	**LECIE:** 上次错误号中断使能(Last error code interrupt enable) 0：当检测到错误，硬件设置 LEC[2:0]时，不设置 ERRI 位； 1：当检测到错误，硬件设置 LEC[2:0]时，设置 ERRI 位为'1'。
位 10	**BOFIE:** 离线中断使能(Bus-off interrupt enable) 0：当 BOFF 位被置'1'时，不设置 ERRI 位； 1：当 BOFF 位被置'1'时，设置 ERRI 位为'1'。
位 9	**EPVIE:** 错误被动中断使能(Error Passive Interrupt Enable) 0：当 EPVF 位被置'1'时，不设置 ERRI 位； 1：当 EPVF 位被置'1'时，设置 ERRI 位为'1'。

位 8	**EWGIE:** 错误警告中断使能(Error warning interrupt enable) 0：当 EWGF 位被置'1'时，不设置 ERRI 位； 1：当 EWGF 位被置'1'时，设置 ERRI 位为'1'。
位 7	保留位，硬件强制为 0
位 6	**FOVIE1:** FIFO 1 溢出中断使能(FIFO overrun interrupt enable) 0：当 FIFO 1 的 FOVR 位被置'1'时，不产生中断； 1：当 FIFO 1 的 FOVR 位被置'1'时，产生中断。
位 5	**FFIE1:** FIFO 1 满中断使能(FIFO full interrupt enable) 0：当 FIFO 1 的 FULL 位被置'1'时，不产生中断； 1：当 FIFO 1 的 FULL 位被置'1'时，产生中断。
位 4	**FMPIE1:** FIFO 1 消息挂号中断使能(FIFO message pending interrupt enable) 0：当 FIFO 1 的 FMP[1:0]位为非 0 时，不产生中断； 1：当 FIFO 1 的 FMP[1:0]位为非 0 时，产生中断。
位 3	**FOVIE0:** FIFO 0 溢出中断使能(FIFO overrun interrupt enable) 0：当 FIFO 0 的 FOVR 位被置'1'时，不产生中断； 1：当 FIFO 0 的 FOVR 位被置'1'时，产生中断。
位 2	**FFIE0:** FIFO 0 满中断使能(FIFO full interrupt enable) 0：当 FIFO 0 的 FULL 位被置'1'时，不产生中断； 1：当 FIFO 0 的 FULL 位被置'1'时，产生中断。
位 1	**FMPIE0:** FIFO 0 消息挂号中断使能(FIFO message pending interrupt enable) 0：当 FIFO 0 的 FMP[1:0]位为非 0 时，不产生中断； 1：当 FIFO 0 的 FMP[1:0]位为非 0 时，产生中断。
位 0	**TMEIE:** 发送邮箱空中断使能(Transmit mailbox empty interrupt enable) 0：当 RQCPx 位被置'1'时，不产生中断； 1：当 RQCPx 位被置'1'时，产生中断。 注：请参考 19.8 节 bxCAN 中断。

CAN 错误状态寄存器(CAN_ESR)

地址偏移量: 0x18

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REC[7:0]								TEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								LEC[2:0]				保留	BOFF	EPVF	WEGF
								rw	rw	rw			r	r	r

位 31:24	**REC[7:0]:** 接收错误计数器(Receive error counter) 这个计数器按照 CAN 协议的故障界定机制的接收部分实现。按照 CAN 的标准，当接收出错时，根据出错的条件，该计数器加 1 或加 8；而在每次接收成功后，该计数器减 1，或当该计数器的值大于 128 时，设置它的值为 120。当该计数器的值超过 127 时，CAN 进入错误被动状态。
位 23:16	**TEC[7:0]:** 9 位发送错误计数器的低 8 位 (Least significant byte of the 9-bit transmit error counter) 与上面相似，这个计数器按照 CAN 协议的故障界定机制的发送部分实现。
位 15:7	保留位，硬件强制为 0。

位 6:4	**LEC[2:0]:** 上次错误代码(Last error code) 在检测到 CAN 总线上发生错误时，硬件根据出错情况设置。当报文被正确发送或接收后，硬件清除其值为'0'。 硬件没有使用错误代码 7，软件可以设置该值，从而可以检测代码的更新。 000: 没有错误； 001: 位填充错； 010: 格式(Form)错； 011: 确认(ACK)错； 100: 隐性位错； 101: 显性位错； 110: CRC 错； 111: 由软件设置。
位 3	保留位，硬件强制为 0。
位 2	**BOFF:** 离线标志(Bus-off flag) 当进入离线状态时，硬件对该位置'1'。当发送错误计数器 TEC 溢出，即大于 255 时，CAN 进入 离线状态。请参考 19.7.6。
位 1	**EPVF:** 错误被动标志(Error passive flag) 当出错次数达到错误被动的阈值时，硬件对该位置'1'。 (接收错误计数器或发送错误计数器的值>127)。
位 0	**EWGF:** 错误警告标志(Error warning flag) 当出错次数达到警告的阈值时，硬件对该位置'1'。 (接收错误计数器或发送错误计数器的值≥96)。

CAN 位时序寄存器(CAN_BTR)

地址偏移量: 0x1C

复位值: 0x0123 0000

注：　当 CAN 处于初始化模式时，该寄存器只能由软件访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SILM	LBKM	保留				SJW[1:0]		保留	TS2[2:0]			TS1[3:0]			
rw	rw	res				rw	rw	res	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						BRP[9:0]									
res						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31	**SILM**: 静默模式(用于调试) (Silent mode (debug)) 0: 正常状态； 1: 静默模式。
位 30	**LBKM**: 环回模式(用于调试) (Loop back mode (debug)) 0: 禁止环回模式； 1: 允许环回模式。
位 29:26	保留位，硬件强制为 0。
位 25:24	**SJW[1:0]:** 重新同步跳跃宽度(Resynchronization jump width) 为了重新同步，该位域定义了 CAN 硬件在每位中可以延长或缩短多少个时间单元的上限。 t_{RJW} = t_{CAN} x (SJW[1:0] + 1)。
位 23	保留位，硬件强制为 0。
位 22:20	**TS2[2:0]:** 时间段 2 (Time segment 2) 该位域定义了时间段 2 占用了多少个时间单元 t_{BS2} = t_{CAN} x (TS2[2:0] + 1)。
位 19:16	**TS1[3:0]:** 时间段 1 (Time segment 1) 该位域定义了时间段 1 占用了多少个时间单元 t_{BS1} = t_{CAN} x (TS1[3:0] + 1) 关于位时间特性的详细信息，请参考 19.7.7 节位
位 15:10	保留位，硬件强制其值为 0。
位 9:0	**BRP[9:0]:** 波特率分频器(Baud rate prescaler) 该位域定义了时间单元(t_q)的时间长度 t_q = (BRP[9:0]+1) x t_{PCLK}

18.9.3 CAN 邮箱寄存器

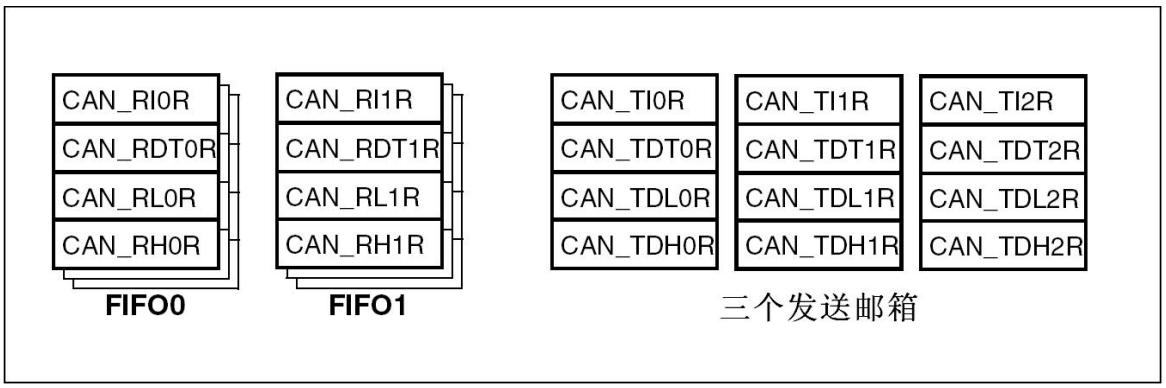
本节描述发送和接收邮箱寄存器。关于寄存器映像的详细信息，请参考 19.7.5 节报文存储。

除了下述例外，发送和接收邮箱几乎一样：

- CAN_RDTxR 寄存器的 FMI 域；
- 接收邮箱是只读的；
- 发送邮箱只有在它为空时才是可写的，CAN_TSR 寄存器的相应 TME 位为'1'，表示发送邮箱为空。

共有 3 个发送邮箱和 2 个接收邮箱。每个接收邮箱为 3 级深度的 FIFO，并且只能访问 FIFO 中最先收到的报文。

每个邮箱包含 4 个寄存器。



发送邮箱标识符寄存器(CAN_TIxR) (x=0..2)

地址偏移量: 0x180，0x190，0x1A0

复位值: 0xXXXX XXXX，X=未定义位(除了第 0 位，复位时 TXRQ=0)

注： *1 当其所属的邮箱处在等待发送的状态时，该寄存器是写保护的*

2 该寄存器实现了发送请求控制功能(第 0 位)－复位值为 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STID[10:0]/EXID[28:18]											EXID[17:13]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXID[12:0]													IDE	RTR	TXRQ
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:21	**STID[10:0]/EXID[28:18]:** 标准标识符或扩展标识符 (Standard identifier or extended identifier) 依据 IDE 位的内容，这些位或是标准标识符，或是扩展身份标识的高字节。
位 20:3	**EXID[17:0]:** 扩展标识符(Extended identifier) 扩展身份标识的低字节。

位 2	**IDE:** 标识符选择(Identifier extension) 该位决定发送邮箱中报文使用的标识符类型 0：使用标准标识符； 1：使用扩展标识符。
位 1	**RTR:** 远程发送请求(Remote transmission request) 0：数据帧； 1：远程帧。
位 0	**TXRQ:** 发送数据请求(Transmit mailbox request) 由软件对其置'1'，来请求发送邮箱的数据。当数据发送完成，邮箱为空时，硬件对其清'0'。

发送邮箱数据长度和时间戳寄存器 (CAN_TDTxR) (x=0..2)

当邮箱不在空置状态时，该寄存器的所有位为写保护。

地址偏移量：0x184，0x194，0x1A4

复位值：未定义

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留							TGT	保留				DLC[3:0]			
res							rw	res				rw	rw	rw	rw

位 31:16	**TIME[15:0]:** 报文时间戳(Message time stamp) 该域包含了，在发送该报文 SOF 的时刻，16 位定时器的值。
位 15:9	保留位
位 8	**TGT:** 发送时间戳(Transmit global time) 只有在 CAN 处于时间触发通信模式，即 CAN_MCR 寄存器的 TTCM 位为'1'时，该位才有效。 0：不发送时间戳 TIME[15:0]； 1：发送时间戳 TIME[15:0]。在长度为 8 的报文中，时间戳 TIME[15:0]是最后 2 个发送的字节：TIME[7:0]作为第 7 个字节，TIME[15:8]为第 8 个字节，它们替换了写入 CAN_TDHxR[31:16]的数据(DATA6[7:0]和 DATA7[7:0])。为了把时间戳的 2 个字节发送出去，DLC 必须编程为 8。
位 7:4	保留位。
位 3:0	**DLC[15:0]:** 发送数据长度(Data length code) 该域指定了数据报文的数据长度或者远程帧请求的数据长度。1 个报文包含 0 到 8 个字节数据，而这由 DLC 决定。

发送邮箱低字节数据寄存器（CAN_TDLxR）（x=0..2）

当邮箱不在空置状态时，该寄存器的所有位为写保护。

地址偏移量：0x188，0x198，0x1A8

复位值：未定义位

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[7:0]								DATA0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24	**DATA3[7:0]：** 数据字节 3 (Data byte 3) 报文的数据字节 3。
位 23:16	**DATA2[7:0]：** 数据字节 2 (Data byte 2) 报文的数据字节 2。

位 15:8	**DATA1[7:0]：** 数据字节 1 (Data byte 1) 报文的数据字节 1。
位 7:0	**DATA0[7:0]：** 数据字节 0 (Data byte 0) 报文的数据字节 0。 报文包含 0 到 8 个字节数据，且从字节 0 开始。

发送邮箱高字节数据寄存器（CAN_TDHxR）（x=0..2）

当邮箱不在空置状态时，该寄存器的所有位为写保护。

地址偏移量：0x18C，0x19C，0x1AC

复位值：未定义位

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA5[7:0]								DATA4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

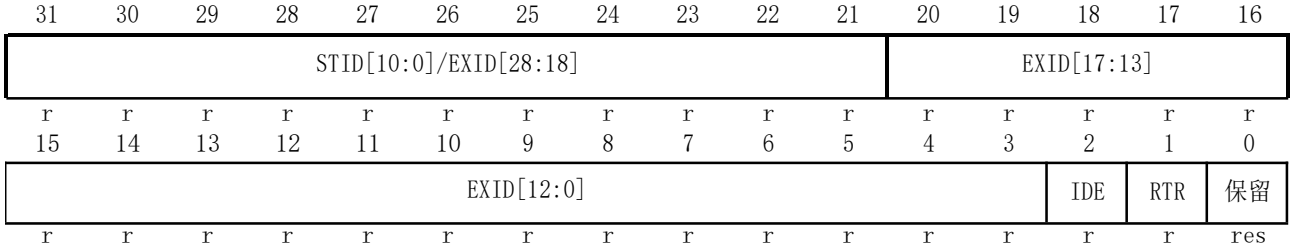
位 31:24	**DATA7[7:0]：** 数据字节 7 (Data byte 7) 报文的数据字节 7 注： 如果 CAN_MCR 寄存器的 TTCM 位为'1'，且该邮箱的 TGT 位也为'1'，那么 DATA7 和 DATA6 将被 TIME 时间戳代替。
位 23:16	**DATA6[7:0]：** 数据字节 6 (Data byte 6) 报文的数据字节 6。
位 15:8	**DATA5[7:0]：** 数据字节 5 (Data byte 5) 报文的数据字节 5。
位 7:0	**DATA4[7:0]：** 数据字节 4 (Data byte 4) 报文的数据字节 4。

接收 FIFO 邮箱标识符寄存器(CAN_RIxR) (x=0..1)

地址偏移量：0x1B0，0x1C0

复位值：未定义位

注：所有接收邮箱寄存器都是只读的。



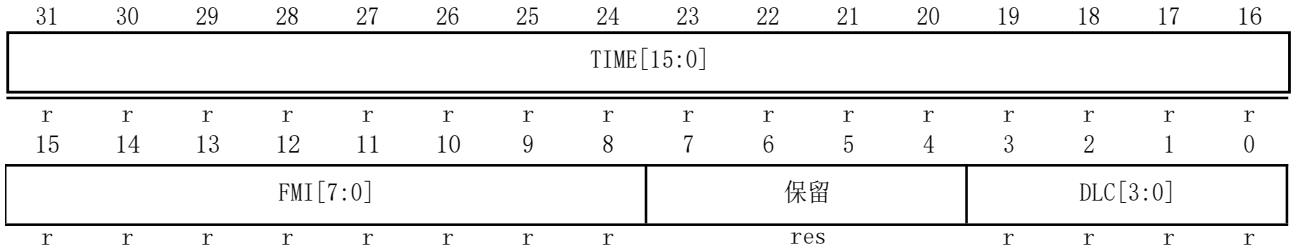
位 31:21	**STID[10:0]/EXID[28:18]:** 标准标识符或扩展标识符 (Standard identifier or extended identifier) 依据 IDE 位的内容，这些位或是标准标识符，或是扩展身份标识的高字节。
位 20:3	**EXID[17:0]:** 扩展标识符(Extended identifier) 扩展标识符的低字节。
位 2	**IDE:** 标识符选择(Identifier extension) 该位决定接收邮箱中报文使用的标识符类型 0：使用标准标识符； 1：使用扩展标识符。
位 1	**RTR:** 远程发送请求(Remote transmission request) 0：数据帧； 1：远程帧。
位 0	保留位。

接收 FIFO 邮箱数据长度和时间戳寄存器(CAN_RDTxR) (x=0..1)

地址偏移量：0x1B4，0x1C4

复位值：未定义

注： 所有接收邮箱寄存器都是只读的。



位 31:16	**TIME[15:0]:** 报文时间戳(Message time stamp) 该域包含了，在接收该报文 SOF 的时刻，16 位定时器的值。
位 15:8	**FMI[15:0]:** 过滤器匹配序号(Filter match index) 这里是存在邮箱中的信息传送的过滤器序号。关于标识符过滤的细节，请参考 19.7.4 中有关过滤器匹配序号。
位 7:4	保留位，硬件强制为 0。
位 3:0	**DLC[15:0]:** 接收数据长度(Data length code) 该域表明接收数据帧的数据长度(0～8)。对于远程帧请求，数据长度 DLC 恒为 0。

接收 FIFO 邮箱低字节数据寄存器(CAN_RDLxR) (x=0..1)

地址偏移量：0x1B8，0x1C8

复位值：未定义位

注：所有接收邮箱寄存器都是只读的。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[7:0]								DATA0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:24	**DATA3[7:0]：** 数据字节 3 (Data byte 3) 报文的数据字节 3。
位 23:16	**DATA2[7:0]：** 数据字节 2 (Data byte 2) 报文的数据字节 2。
位 15:8	**DATA1[7:0]：** 数据字节 1 (Data byte 1) 报文的数据字节 1。
位 7:0	**DATA0[7:0]：** 数据字节 0 (Data byte 0) 报文的数据字节 0。 报文包含 0 到 8 个字节数据，且从字节 0 开始。

接收 FIFO 邮箱高字节数据寄存器(CAN_RDHxR) (x=0..1)

地址偏移量：0x1BC，0x1CC

复位值：未定义位

注：所有接收邮箱寄存器都是只读的。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA5[7:0]								DATA4[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:24	**DATA7[7:0]：** 数据字节 7 (Data byte 7) 报文的数据字节 7
位 23:16	**DATA6[7:0]：** 数据字节 6 (Data byte 6) 报文的数据字节 6。
位 15:8	**DATA5[7:0]：** 数据字节 5 (Data byte 5) 报文的数据字节 5。
位 7:0	**DATA4[7:0]：** 数据字节 4 (Data byte 4) 报文的数据字节 4。

18.9.4 CAN 过滤器寄存器

CAN 过滤器主控寄存器(CAN_FMR)

地址偏移量: 0x200

复位值: 0x2A1C 0E01

注： 该寄存器的非保留位完全由软件控制。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				保留				保留							FINIT
res				rw				res							rw

位 31:14	保留位，强制为复位值。
位 13:8	保留位，强制为复位值。
位 7:1	保留位，强制为复位值。
位 0	**FINIT：** 过滤器初始化模式(Filter init mode) 针对所有过滤器组的初始化模式设置。 0: 过滤器组工作在正常模式； 1: 过滤器组工作在初始化模式。

CAN 过滤器模式寄存器(CAN_FM1R)

地址偏移量: 0x204

复位值: 0x0000 0000

注： 只有在设置 CAN_FMR(FINIT=1)，使过滤器处于初始化模式下，才能对该寄存器写入。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		FBM13	FBM12	FBM11	FBM10	FBM9	FBM8	FBM7	FBM6	FBM5	FBM4	FBM3	FBM2	FBM1	FBM0
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

注：请参考图 147：过滤器组位宽设置－寄存器组织。

位 31:28	保留位，硬件强制为 0
位 13:0	**FBMx：** 过滤器模式(Filter mode) 过滤器组 x 的工作模式。 0: 过滤器组 x 的 2 个 32 位寄存器工作在标识符屏蔽位模式； 1: 过滤器组 x 的 2 个 32 位寄存器工作在标识符列表模式。

CAN 过滤器位宽寄存器(CAN_FS1R)

地址偏移量: 0x20C

复位值: 0x0000 0000

注： 只有在设置 CAN_FMR(FINIT=1)，使过滤器处于初始化模式下，才能对该寄存器写入。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		FSC13	FSC12	FSC11	FSC10	FSC9	FSC8	FSC7	FSC6	FSC5	FSC4	FSC3	FSC2	FSC1	FSC0
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

注：请参考图 147：过滤器组位宽设置－寄存器组织。

位 31:28	保留位，硬件强制为 0
位 13:0	**FSCx：** 过滤器位宽设置(Filter scale configuration) 过滤器组 x(13～0)的位宽。 0：过滤器位宽为 2 个 16 位； 1：过滤器位宽为单个 32 位。

CAN 过滤器 FIFO 关联寄存器(CAN_FFA1R)

地址偏移量: 0x214

复位值: 0x0000 0000

注： 只有在设置 CAN_FMR(FINIT=1)，使过滤器处于初始化模式下，才能对该寄存器写入。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		FFA13	FFA12	FFA11	FFA10	FFA9	FFA8	FFA7	FFA6	FFA5	FFA4	FFA3	FFA2	FFA1	FFA0
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:14	保留位，硬件强制为 0。
位 13:0	**FFAx：** 过滤器位宽设置(Filter FIFO assignment for filter x) 报文在通过了某过滤器的过滤后，将被存放到其关联的 FIFO 中。 0：过滤器被关联到 FIFO0； 1：过滤器被关联到 FIFO1。

CAN 过滤器激活寄存器(CAN_FA1R)

地址偏移量: 0x21C

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		FACT13	FACT12	FACT11	FACT10	FACT9	FACT8	FACT7	FACT6	FACT5	FACT4	FACT3	FACT2	FACT1	FACT0
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:14	保留位，硬件强制为 0。
位 13:0	**FACTx** ： 过滤器激活(Filter active) 软件对某位设置'1'来激活相应的过滤器。只有对 FACTx 位清'0'，或对 CAN_FMR 寄存器的 FINIT 位设置'1'后，才能修改相应的过滤器寄存器 x(CAN_FxR[0:1])。 0：过滤器被禁用； 1：过滤器被激活。

CAN 过滤器组 i 的寄存器 x （CAN_FiRx）（i＝0..13； x=1..2）

地址偏移量：0x240h..0x31C

复位值：未定义位

注： *共有 14 组过滤器：i=0..13。每组过滤器由 2 个 32 位的寄存器，CAN_FiR[2:1]组成。只有在 CAN_FAxR 寄存器相应的 FACTx 位清'0'，或 CAN_FMR 寄存器的 FINIT 位为'1'时，才能修改相应的过滤器寄存器。*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		FB13	FB12	FB11	FB10	FB9	FB8	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

在所有的配置情况下：

位 13:0	**FB[13:0]**：过滤器位(Filter bits) **标识符模式** 寄存器的每位对应于所期望的标识符的相应位的电平。 0: 期望相应位为显性位； 1: 期望相应位为隐性位。 **屏蔽位模式** 寄存器的每位指示是否对应的标识符寄存器位一定要与期望的标识符的相应位一致。 0: 不关心，该位不用于比较； 1: 必须匹配，到来的标识符位必须与滤波器对应的标识

注： *根据过滤器位宽和模式的不同设置，过滤器组中的两个寄存器的功能也不尽相同。关于过滤器的映射，功能描述和屏蔽寄存器的关联，请参见 19.7.4 节标识符过滤。*

屏蔽位模式下的屏蔽/标识符寄存器，跟**标识符列表模式**下的寄存器位定义相同。 关于过滤器组寄存器的地址，请参见表 75。

18.9.5 bxCAN 寄存器列表

关于寄存器的起始地址，参见表 1。

表 75 bxCAN－寄存器列表及其复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	CAN_MCR	保留															DBF	RESET	保留							TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
	复位值																1	0								0	0	0	0	0	0	1	0
004h	CAN_MSR	保留																				RX	SAMP	RXM	TXM	保留			SLAKI	WKUI	ERRI	SLAK	INAK
	复位值																					1	1	0	0				0	0	0	1	0
008h	CAN_TSR	LOW[2:0]			TME[2:0]			CODE[1:0]		ABRQ2	保留			TERR2	ALST2	TXOK2	RQCP2	ABRQ1	保留			TERR1	ALST1	TXOK1	RQCP1	ABRQ0	保留			TERR0	ALST0	TXOK0	RQCP0
	复位值	0	0	0	1	1	1	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0
00Ch	CAN_RFOR	保留																										RFOM0	FOVR0	FULL0	保留	FMP0[1:0]	
	复位值																											0	0	0		0	0
010h	CAN_RF1F	保留																										RFOM1	FOVR1	FULL1	保留	FMP1[1:0]	
	复位值																											0	0	0		0	0
014h	CAN_IER	保留														SLKIE	WKUIE	ERRIE	保留			LECIE	BOFIE	EPVIE	EWGIE	保留	FOVIE	FFIE1	FMPIE1	FOVIE0	FFIE0	FMPIE0	TMEIE
	复位值															0	0	0				0	0	0	0		0	0	0	0	0	0	0
018h	CAN_ESR	REC[7:0]								TEC[7:0]									保留								LEC[2:0]			保留	BOFF	EPVF	EWGF
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									0	0	0		0	0	0
01Ch	CAN_BTR	SILM	LBKM	保留					SJW[1:0]		保留	TS2[2:0]			TS1[3:0]				保留								BRP[9:0]							
	复位值	0	0						0	0		0	1	0	0	0	1	1									0	0	0	0	0	0	0	0	0	0
020h~17Fh	保留																																
180h	CAN_TIOR	STID[10:0]/EXUD[28:18]											EXID[17:0]																		IDE	RTR	TXRQ
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
184h	CAN_TDTOR	TIME[15:0]																	保留							TGT	保留				DLC[3:0]			
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
188h	CAN_TDLOR	DATA3[7:0]								DATA2[7:0]									DATA1[7:0]								DATA0[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
18Ch	CAN_TDHOR	DATA7[7:0]								DATA6[7:0]									DATA5[7:0]								DATA4[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
190h	CAN_TI1R	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	TXRQ
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
194h	CAN_TDT1R	TIME[15:0]																保留							TGT	保留					DLC[3:0]		
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x						x	x	x	x
198h	CAN_TDL1R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
19Ch	CAN_TDH1R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1A0h	CAN_TI2R	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	TXR
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
1A4h	CAN_TDT2R	TIME[15:0]																保留							TGT	保留					DLC[3:0]		
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x						x	x	x	x
1A8h	CAN_TDL2R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1ACh	CAN_TDH2R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1B0h	CAN_RI0R	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	保留
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1B4h	CAN_RDT0R	TIME[15:0]																FMI[7:0]								保留					DLC[3:0]		
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x						x	x	x	x
1B8h	CAN_RDL0R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1BCh	CAN_RDH0R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1C0h	CAN_RI1R	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	保留
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1C4h	CAN_RDT1R	TIME[15:0]																FMI[7:0]								保留					DLC[3:0]		
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x						x	x	x	x
1C8h	CAN_RDL1R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1CCh	CAN_RDH1R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1D0h~1FFh																																	

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
200h	CAN_FMR	保留																		保留						保留							FINIT
	复位值																																1
204h	CAN_FM1R	保留																		FBM[27:0]													
	复位值																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
208h	保留																																
20Ch	CAN_FS1R	保留																		FSC[27:0]													
	复位值																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
210h	保留																																
214h	CAN_FFA1R	保留																		FFA[27:0]													
	复位值																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
218h	保留																																
21Ch	CAN_FA1R	保留																		FACT[27:0]													
	复位值																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
220h																																	
224~ 23Fh																																	
240h	CAN_FOR1	FB[13:0]																															
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
244h	CAN_FOR2	FB[13:0]																															
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
248h	CAN_F1R1	FB[13:0]																															
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
24Ch	CAN_F1R2	FB[13:0]																															
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
. . .	. . .	...																															
318h	CAN_F27R1	FB[13:0]																															
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
31Ch	CAN_F27R2	FB[13:0]																															
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

19 串行外设接口（SPI）

19.1 SPI 简介

19.2 SPI 主要特征

19.2.1 SPI 特征

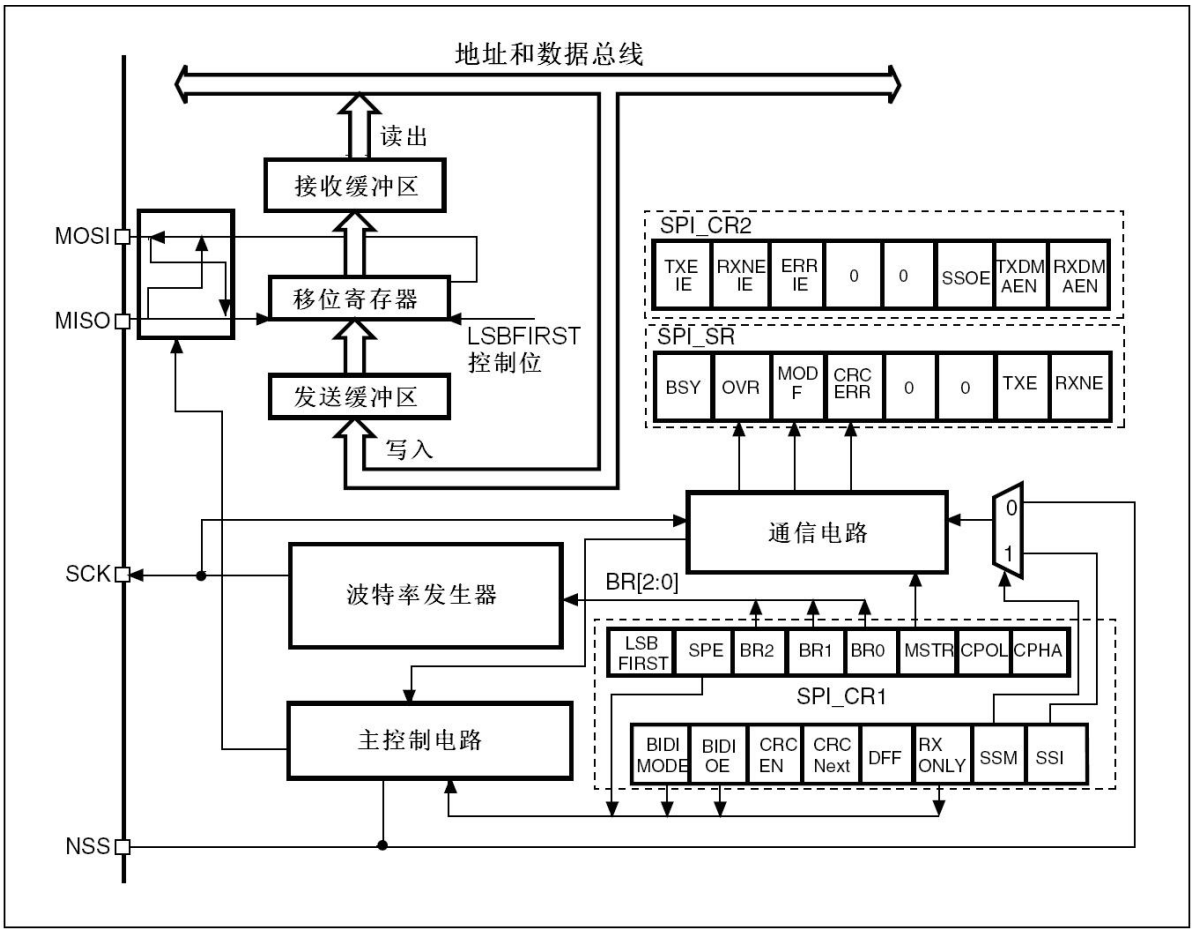
- 3 线全双工同步传输
- 带或不带第三根双向数据线的双线单工同步传输
- 8 或 16 位传输帧格式选择
- 主或从操作
- 支持多主模式
- 8 个主模式波特率预分频系数(最大为 $f_{PCLK}/2$)
- 从模式频率(最大为 $f_{PCLK}/2$)
- 主模式和从模式的快速通信
- 主模式和从模式下均可以由软件或硬件进行 NSS 管理：主/从操作模式的动态改变
- 可编程的时钟极性和相位
- 可编程的数据顺序，MSB 在前或 LSB 在前
- 可触发中断的专用发送和接收标志
- SPI 总线忙状态标志
- 支持可靠通信的硬件 CRC
  - 在发送模式下，CRC 值可以被作为最后一个字节发送
  - 在全双工模式中对接收到的最后一个字节自动进行 CRC 校验
- 可触发中断的主模式故障、过载以及 CRC 错误标志
- 支持 DMA 功能的 1 字节发送和接收缓冲器：产生发送和接受请求
- HK32F103 的 SPI 接口支持 2~32 比特数据包长度可配（此功能无法与 SPI 的 CRC 校验功能同时使用）

19.3 SPI 功能描述

19.3.1 概述

SPI 的方框图见下图。

图 154 SPI 框图

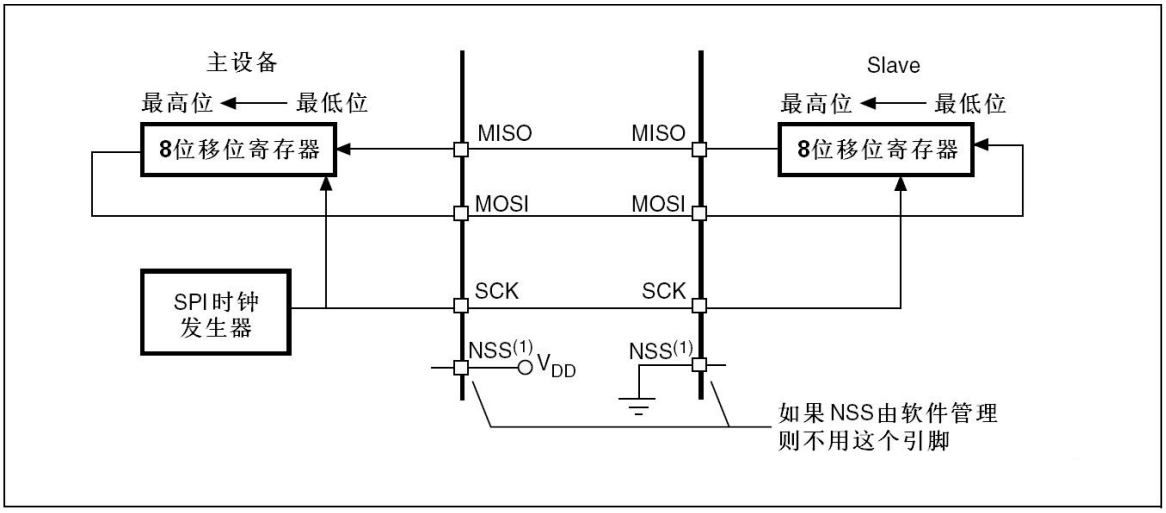


通常 SPI 通过 4 个引脚与外部器件相连：

- MISO：主设备输入/从设备输出引脚。该引脚在从模式下发送数据，在主模式下接收数据。
- MOSI：主设备输出/从设备输入引脚。该引脚在主模式下发送数据，在从模式下接收数据。
- SCK：串口时钟，作为主设备的输出，从设备的输入
- NSS：从设备选择。这是一个可选的引脚，用来选择主/从设备。它的功能是用来作为“片 选引脚”，让主设备可以单独地与特定从设备通讯，避免数据线上的冲突。从设备的 NSS 引脚可以由主设备的一个标准 I/O 引脚来驱动。一旦被使能(SSOE 位),NSS 引脚也可以作为 输出引脚，并在 SPI 处于主模式时拉低；此时，所有的 SPI 设备，如果它们的 NSS 引脚连接 到主设备的 NSS 引脚，则会检测到低电平，如果它们被设置为 NSS 硬件模式，就会自动进 入从设备状态。当配置为主设备、NSS 配置为输入引脚(MSTR=1，SSOE=0)时，如果 NSS 被拉低，则这个 SPI 设备进入主模式失败状态：即 MSTR 位被自动清除，此设备进入从模式 。

下图是一个单主和单从设备互连的例子。

图 155 单主和单从应用



1. 这里 NSS 引脚设置为输入

MOSI 脚相互连接，MISO 脚相互连接。这样，数据在主和从之间串行地传输(MSB 位在前)。

通信总是由主设备发起。主设备通过 MOSI 脚把数据发送给从设备，从设备通过 MISO 引脚回传数据。这意味全双工通信的数据输出和数据输入是用同一个时钟信号同步的；时钟信号由主设备通过 SCK 脚提供。

从设备选择（NSS）脚管理

可以通过 SPI_CR1 的 SSM 位设置硬件或者软件管理从设备选择：

- 软件 NSS 模式（SSM=1）：NSS 信号电平通过写内部 SPI_CR1 的 SSI 位来控制，在这种模式下 SPI 不需要占用外部 NSS 引脚， 可以用作其他功能。
- 硬件 NSS 模式，基于 NSS 的输出配置（SPI_CR2 的 SSOE 位）有两种情况：
  - NSS 输出使能（SSM=0，SSOE=1）：这种配置只能用于 SPI 主设备，并且在主设备开始传输后 NSS 被驱动为低，并且一直保持到 SPI 关闭。
  - NSS 输出关闭（SSM=0，SSOE=0）：这种配置可以用于主机模式下实现多主机系统。在从机模式下，这种配置就是经典的 NSS 引脚作为从机输入信号，当 NSS 为低时从机被选中，当 NSS 为高时从机未被选中

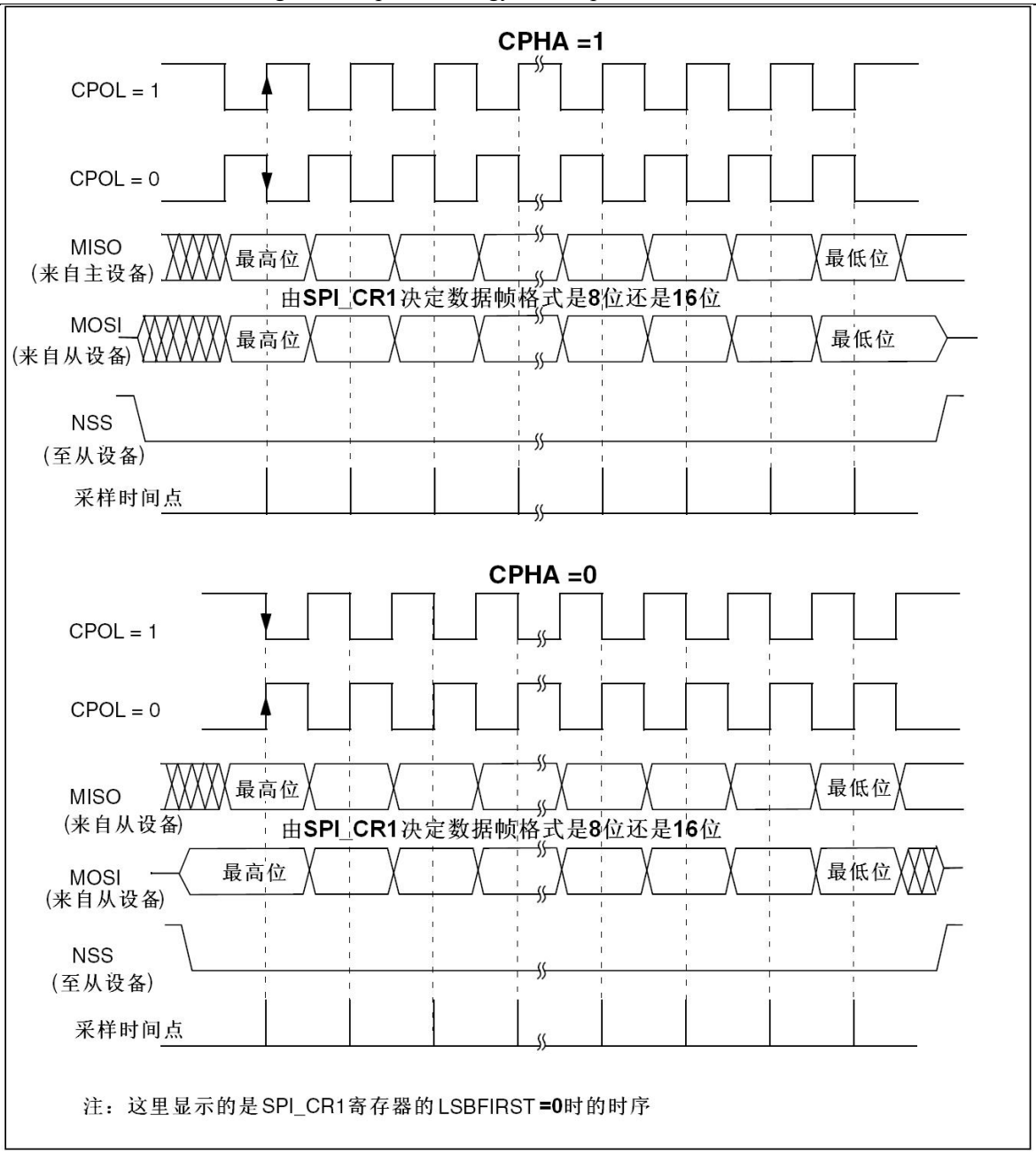
时钟信号的相位和极性

SPI_CR 寄存器的 CPOL 和 CPHA 位，能够组合成四种可能的时序关系。CPOL(时钟极性)位控制在没有数据传输时时钟的空闲状态电平，此位对主模式和从模式下的设备都有效。如果 CPOL 被清'0'，SCK 引脚在空闲状态保持低电平；如果 CPOL 被置'1'，SCK 引脚在空闲状态保持高电平。如果 CPHA(时钟相位)位被置'1'，SCK 时钟的第二个边沿(CPOL 位为 0 时就是下降沿，CPOL 位为'1'时就是上升沿)进行数据位的采样，数据在第二个时钟边沿被锁存。如果 CPHA 位被清'0'，SCK 时钟的第一边沿(CPOL 位为'0'时就是上升沿，CPOL 位为'1'时就是下降沿)进行数据位采样，数据在第一个时钟边沿被锁存。

CPOL 时钟极性和 CPHA 时钟相位的组合选择数据捕捉的时钟边沿。 图 157 显示了 SPI 传输的 4 种 CPHA 和 CPOL 位组合。此图可以解释为主设备和从设备的 SCK 脚、MISO 脚、MOSI 脚直接连接的主或从时序图。

注意：
1. *在改变 CPOL/CPHA 位之前，必须清除 SPE 位将 SPI 禁止。*
2. *主和从必须配置成相同的时序模式。*
3. *SCK 的空闲状态必须和 SPI_CR1 寄存器指定的极性一致(CPOL 为'1'时，空闲时应上拉 SCK 为高电平；CPOL 为'0'时，空闲时应下拉 SCK 为低电平)。*
4. *数据帧格式(8 位或 16 位)由 SPI_CR1 寄存器的 DFF 位选择，并且决定发送/接收的数据长度。*

图 157 数据时钟时序图



注：这里显示的是SPI_CR1寄存器的LSBFIRST=0时的时序

数据帧格式

根据 SPI_CR1 寄存器中的 LSBFIRST 位，输出数据位时可以 MSB 在先也可以 LSB 在先。 根据 SPI_CR1 寄存器的 DFF 位，每个数据帧可以是 8 位或是 16 位。所选择的数据帧格式对发送和/或接收都有效。

19.3.2 配置 SPI 为从模式

在从模式下，SCK 引脚用于接收从主设备来的串行时钟。SPI_CR1 寄存器中 BR[2:0]的设置不影响数据传输速率。

注：建议在主设备发送时钟之前使能 SPI 从设备，否则可能会发生意外的数据传输。在通信时钟的第 一个边沿到来之前或正在进行的通信结束之前，从设备的数据寄存器必须就绪。在使能从设备 和主设备之前，通信时钟的极性必须处于稳定的数值。

请按照以下步骤配置 SPI 为从模式：

配置步骤

1. 设置 DFF 位以定义数据帧格式为 8 位或 16 位。
2. 选择 CPOL 和 CPHA 位来定义数据传输和串行时钟之间的相位关系(见图 157)。为保证正确的数据传输，从设备和主设备的 CPOL 和 CPHA 位必须配置成相同的方式。
3. 帧格式(SPI_CR1 寄存器中的 LSBFIRST 位定义的”MSB 在前”还是”LSB 在前”)必须与主设备相同。
4. 硬件模式下(参考从选择(NSS)脚管理部分)，在完整的数据帧(8 位或 16 位)传输过程中，NSS 引脚必须为低电平。在 NSS 软件模式下，设置 SPI_CR1 寄存器中的 SSM 位并清除 SSI 位。
5. 清除 MSTR 位、设置 SPE 位(SPI_CR1 寄存器)，使相应引脚工作于 SPI 模式下。 在这个配置中，MOSI 引脚是数据输入，MISO 引脚是数据输出。

数据发送过程

在写操作中，数据字被并行地写入发送缓冲器。

当从设备收到时钟信号，并且在 MOSI 引脚上出现第一个数据位时，发送过程开始(译注：此时第一个位被发送出去)。余下的位(对于 8 位数据帧格式，还有 7 位；对于 16 位数据帧格式，还有15 位)被装进移位寄存器。当发送缓冲器中的数据传输到移位寄存器时，SPI_SP 寄存器的 TXE 标志被设置，如果设置了 SPI_CR2 寄存器的 TXEIE 位，将会产生中断。

数据接收过程

对于接收器，当数据接收完成时：

- 移位寄存器中的数据传送到接收缓冲器，SPI_SR 寄存器中的 RXNE 标志被设置。
- 如果设置了 SPI_CR2 寄存器中的 RXNEIE 位，则产生中断。

在最后一个采样时钟边沿后，RXNE 位被置’1’，移位寄存器中接收到的数据字节被传送到接收缓冲器。当读 SPI_DR 寄存器时，SPI 设备返回这个接收缓冲器的数值。

读 SPI_DR 寄存器时，RXNE 位被清除。

19.3.3 配置 SPI 为主模式

在主配置时，在 SCK 脚产生串行时钟。

配置步骤

1. 通过 SPI_CR1 寄存器的 BR[2:0]位定义串行时钟波特率。
2. 选择 CPOL 和 CPHA 位，定义数据传输和串行时钟间的相位关系(见图 157)。
3. 设置 DFF 位来定义 8 位或 16 位数据帧格式。
4. 配置 SPI_CR1 寄存器的 LSBFIRST 位定义帧格式。
5. 如果需要 NSS 引脚工作在输入模式，硬件模式下，在整个数据帧传输期间应把 NSS 脚连接 到高电平；在软件模式下，需设置 SPI_CR1 寄存器的 SSM 位和 SSI 位。如果 NSS 引脚工作 在输出模式，则只需设置 SSOE 位。
6. 必须设置 MSTR 位和 SPE 位(只当 NSS 脚被连到高电平，这些位才能保持置位)。 在这个

配置中，MOSI 引脚是数据输出，而 MISO 引脚是数据输入。

数据发送过程

当写入数据至发送缓冲器时，发送过程开始。在发送第一个数据位时，数据字被并行地(通过内部总线)传入移位寄存器，而后串行地移出到 MOSI 脚上；MSB 在先还是 LSB 在先，取决于 SPI_CR1 寄存器中的 LSBFIRST 位的设置。数据从发送缓冲器传输到移位寄存器时 TXE 标志将被置位，如果设置了 SPI_CR1 寄存器中的 TXEIE 位，将产生中断。

数据接收过程

对于接收器来说，当数据传输完成时：

- 传送移位寄存器里的数据到接收缓冲器，并且 RXNE 标志被置位。
- 如果设置了 SPI_CR2 寄存器中的 RXNEIE 位，则产生中断。

在最后采样时钟沿，RXNE 位被设置，在移位寄存器中接收到的数据字被传送到接收缓冲器。读 SPI_DR 寄存器时，SPI 设备返回接收缓冲器中的数据。

读 SPI_DR 寄存器将清除 RXNE 位。一旦传输开始，如果下一个将发送的数据被放进了发送缓冲器，就可以维持一个连续的传输流。在试图写发送缓冲器之前，需确认 TXE 标志应该为'1'。

注：　当一个主机与多个从机通信时，从机需要在传输的某些时刻不被选中，那么该从机的 NSS 引脚必须被配置成 GPIO，或者使用其他某个 GPIO 用来被软件 toggle。

19.3.4 配置 SPI 为单工通信

SPI 模块能够以两种配置工作于单工方式：

- 1 条时钟线和 1 条双向数据线；
- 1 条时钟线和 1 条数据线(只接收或只发送)；

1 条时钟线和 1 条双向数据线(BIDIMODE=1)

设置 SPI_CR1 寄存器中的 BIDIMODE 位而启用此模式。在这个模式下，SCK 引脚作为时钟，主设备使用 MOSI 引脚而从设备使用 MISO 引脚作为数据通信。传输的方向由 SPI_CR1 寄存器里的 BIDIOE 控制，当这个位是'1'的时候，数据线是输出，否则是输入。

1 条时钟和 1 条单向数据线(BIDIMODE=0)

在这个模式下，SPI 模块可以或者作为只发送，或者作为只接收。

- 只发送模式类似于全双工模式(BIDIMODE=0，RXONLY=0)：数据在发送引脚(主模式时是 MOSI、从模式时是 MISO)上传输，而接收引脚(主模式时是 MISO、从模式时是 MOSI)可以作为通用的 I/O 使用。此时，软件不必理会接收缓冲器中的数据(如果读出数据寄存器，它不包含任何接收数据)。
- 在只接收模式，可以通过设置 SPI_CR2 寄存器的 RXONLY 位而关闭 SPI 的输出功能；此时，发送引脚(主模式时是 MOSI、从模式时是 MISO)被释放，可以作为其它功能使用。

配置并使能 SPI 模块为只接收模式的方式是：

- 在主模式时，一旦使能 SPI，通信立即启动，当清除 SPE 位时立即停止当前的接收。在此模式下，不必读取 BSY 标志，在 SPI 通信期间这个标志始终为'1'。
- 在从模式时，只要 NSS 被拉低(或在 NSS 软件模式时，SSI 位为'0')同时 SCK 有时钟脉冲，SPI 就一直在接收。

19.3.5 数据发送与接收过程

接收与发送缓冲器

在接收时，接收到的数据被存放在一个内部的接收缓冲器中；在发送时，在被发送之前，数据将首先被存放在一个内部的发送缓冲器中。

对 SPI_DR 寄存器的读操作，将返回接收缓冲器的内容；写入 SPI_DR 寄存器的数据将被写入发送缓冲器中。

主模式下开始传输

- 全双工模式(BIDIMODE=0 并且 RXONLY=0)
  - 当写入数据到 SPI_DR 寄存器(发送缓冲器)后，传输开始；
  - 在传送第一位数据的同时，数据被并行地从发送缓冲器传送到 8 位的移位寄存器中，然后按顺序被串行地移位送到 MOSI 引脚上；
  - 与此同时，在 MISO 引脚上接收到的数据，按顺序被串行地移位进入 8 位的移位寄存器中，然后被并行地传送到 SPI_DR 寄存器(接收缓冲器)中。
- 单向的只接收模式(BIDIMODE=0 并且 RXONLY=1)
  - SPE=1 时，传输开始；
  - 只有接收器被激活，在 MISO 引脚上接收到的数据，按顺序被串行地移位进入 8 位的移位寄存器中，然后被并行地传送到 SPI_DR 寄存器(接收缓冲器)中。
- 双向模式，发送时(BIDIMODE=1 并且 BIDIOE=1)
  - 当写入数据到 SPI_DR 寄存器(发送缓冲器)后，传输开始；
  - 在传送第一位数据的同时，数据被并行地从发送缓冲器传送到 8 位的移位寄存器中，然后按顺序被串行地移位送到 MOSI 引脚上；
  - 不接收数据。
- 双向模式，接收时(BIDIMODE=1 并且 BIDIOE=0)
  - SPE=1 并且 BIDIOE=0 时，传输开始；
  - 在 MOSI 引脚上接收到的数据，按顺序被串行地移位进入 8 位的移位寄存器中，然后被并行地传送到 SPI_DR 寄存器(接收缓冲器)中。
  - 不激活发送器，没有数据被串行地送到 MOSI 引脚上。

从模式下开始传输

- 全双工模式(BIDIMODE=0 并且 RXONLY=0)
  - 当从设备接收到时钟信号并且第一个数据位出现在它的 MOSI 时，数据传输开始，随后的数据位依次移动进入移位寄存器；
  - 与此同时，在传输第一个数据位时，发送缓冲器中的数据被并行地传送到 8 位的移位寄存器，随后被串行地发送到 MISO 引脚上。软件必须保证在 SPI 主设备开始数据传输之前在发送寄存器中写入要发送的数据。
- 单向的只接收模式(BIDIMODE=0 并且 RXONLY=1)
  - 当从设备接收到时钟信号并且第一个数据位出现在它的 MOSI 时，数据传输开始，随后数据位依次移动进入移位寄存器；
  - 不启动发送器，没有数据被串行地传送到 MISO 引脚上。
- 双向模式，发送时(BIDIMODE=1 并且 BIDIOE=1)
  - 当从设备接收到时钟信号并且发送缓冲器中的第一个数据位被传送到 MISO 引脚上的时候，数据传输开始；
  - 在第一个数据位被传送到 MISO 引脚上的同时，发送缓冲器中要发送的数据被平行地传送到 8 位的移位寄存器中，随后被串行地发送到 MISO 引脚上。软件必须保证在 SPI 主设备开始数据传输之前在发送寄存器中写入要发送的数据；
  - 不接收数据。
- 双向模式，接收时(BIDIMODE=1 并且 BIDIOE=0)
  - 当从设备接收到时钟信号并且第一个数据位出现在它的 MOSI 时，数据传输开始；
  - 从 MISO 引脚上接收到的数据被串行地传送到 8 位的移位寄存器中，然后被平行地传送到 SPI_DR 寄存器(接收缓冲器)；
  - 不启动发送器，没有数据被串行地传送到 MISO 引脚上。

处理数据的发送与接收

当数据从发送缓冲器传送到移位寄存器时，设置 TXE 标志(发送缓冲器空)，它表示内部的发送缓冲器可以接收下一个数据；如果在 SPI_CR2 寄存器中设置了 TXEIE 位，则此时会产生一个中断；写入 SPI_DR 寄存器即可清除 TXE 位。

注：在写入发送缓冲器之前，软件必须确认 TXE 标志为'1'，否则新的数据会覆盖已经在发送缓冲器中的数据。

在采样时钟的最后一个边沿，当数据被从移位寄存器传送到接收缓冲器时，设置 RXNE 标志(接收缓冲器非空)；它表示数据已经就绪，可以从 SPI_DR 寄存器读出；如果在 SPI_CR2 寄存器中设置了 RXNEIE 位，则此时会产生一个中断；读出 SPI_DR 寄存器即可清除 RXNIE 标志位。

在一些配置中，传输最后一个数据时，可以使用 BSY 标志等待数据传输的结束。

主或从模式下(BIDIMODE=0 并且 RXONLY=0)全双工发送和接收过程模式

软件必须遵循下述过程，发送和接收数据(见图 158 和图 159)：

1. 设置 SPE 位为'1'，使能 SPI 模块；
2. 在 SPI_DR 寄存器中写入第一个要发送的数据，这个操作会清除 TXE 标志；
3. 等待 TXE=1，然后写入第二个要发送的数据。等待 RXNE=1，然后读出 SPI_DR 寄存器并获得第一个接收到的数据，读 SPI_DR 的同时清除了 RXNE 位。重复这些操作，发送后续 的数据同时接收 n-1 个数据；
4. 等待 RXNE=1，然后接收最后一个数据；
5. 等待 TXE=1，在 BSY=0 之后关闭 SPI 模块。 也可以在响应 RXNE 或 TXE 标志的上升沿产生的中断的处理程序中实现这个过程。

图 158 主模式、全双工模式下(BIDIMODE=0 并且 RXONLY=0)连续传输时，TXE/RXNE/BSY 的变化示意图

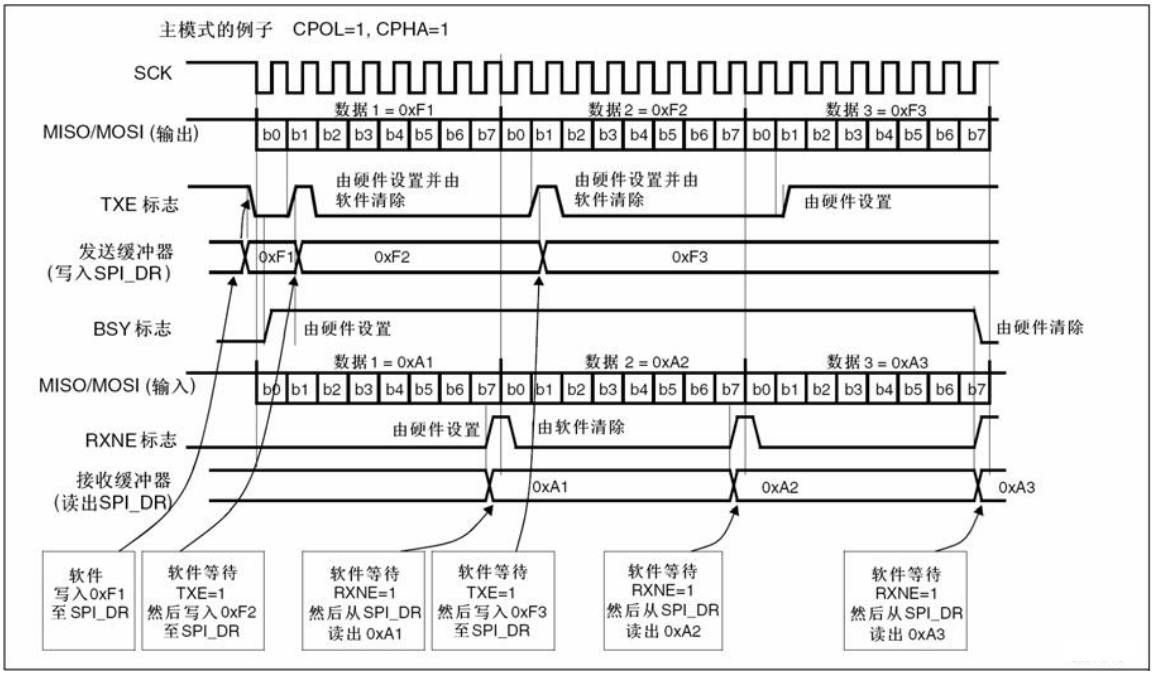
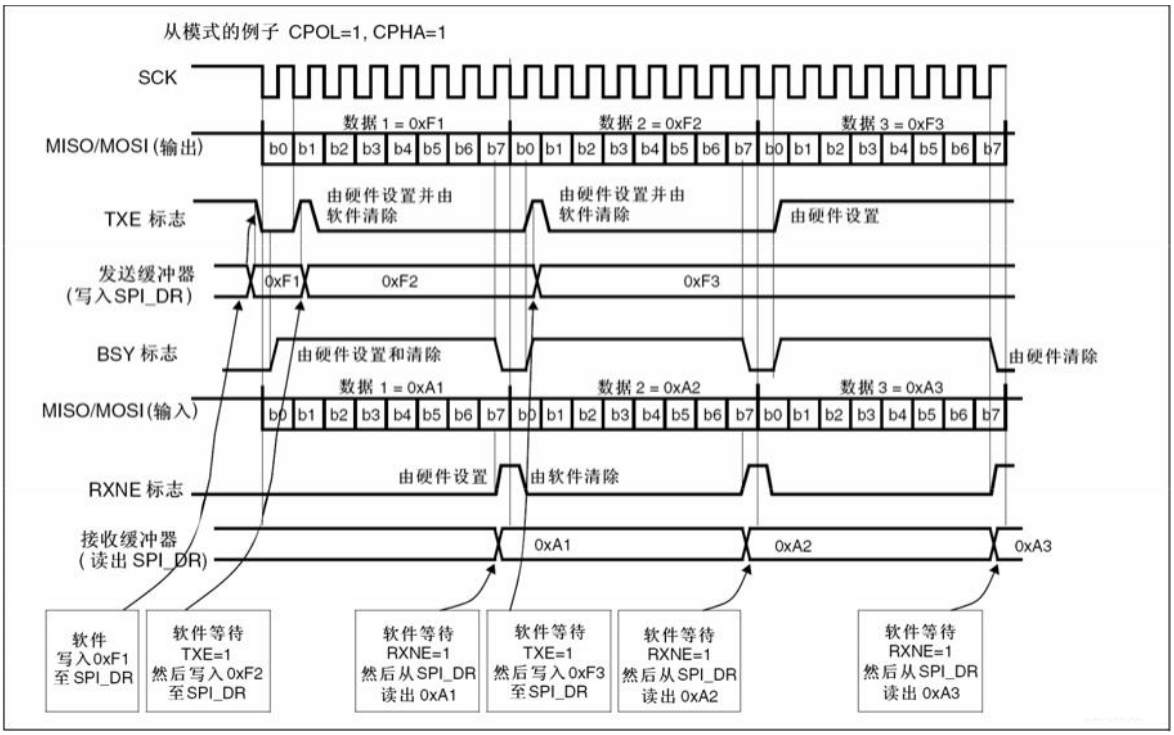


图 159 从模式、全双工模式下(BIDIMODE=0 并且 RXONLY=0)连续传输时，TXE/RXNE/BSY 的变化示意图



只发送过程(BIDIMODE=0 并且 RXONLY=0)

在此模式下，传输过程可以简要说明如下，使用 BSY 位等待传输的结束(见图 160 和图 161)：

1. 设置 SPE 位为'1'，使能 SPI 模块；
2. 在 SPI_DR 寄存器中写入第一个要发送的数据，这个操作会清除 TXE 标志；
3. 等待 TXE=1，然后写入第二个要发送的数据。重复这个操作，发送后续的数据；
4. 写入最后一个数据到 SPI_DR 寄存器之后，等待 TXE=1；然后等待 BSY=0，这表示最后一个数据的传输已经完成。

也可以在响应 TXE 标志的上升沿产生的中断的处理程序中实现这个过程。

注：1. 对于不连续的传输，在写入 SPI_DR 寄存器的操作与设置 BSY 位之间有 2 个 APB 时钟周期的延迟，因此在只发送模式下，写入最后一个数据后，最好先等待 TXE=1，然后再等待 BSY=0。

2. 只发送模式下，在传输 2 个数据之后，由于不会读出接收到的数据，SPI_SR 寄存器中的 OVR 位会变为'1'。(译注：软件不必理会这个 OVR 标志位)

图 160 主设备只发送模式(BIDIMODE=0 并且 RXONLY=0)下连续传输时，,TXE/BSY 变化示意图

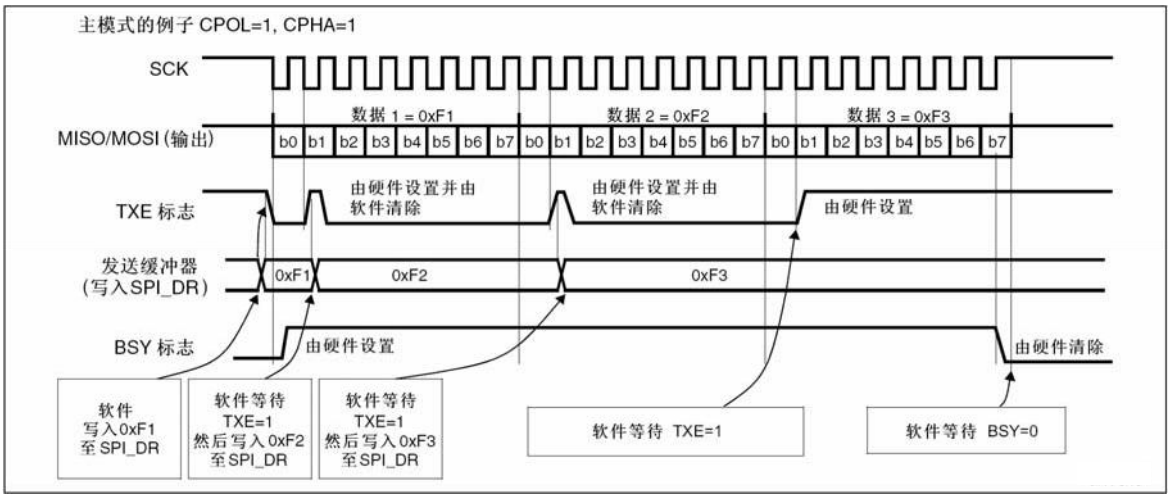
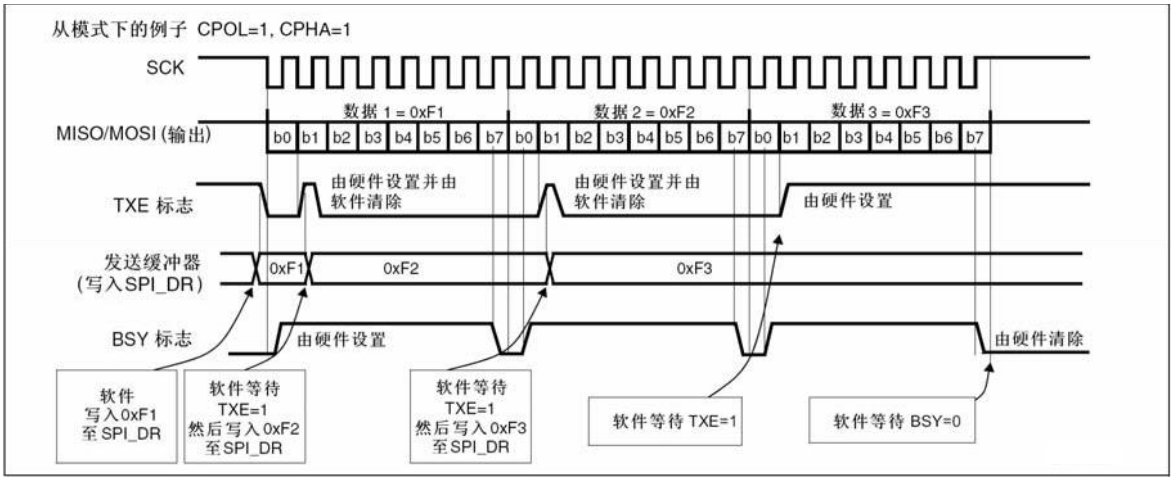


图 161 从设备只发送模式(BIDIMODE=0 并且 RXONLY=0)下连续传输时，TXE/BSY 变化示意图



双向发送过程(BIDIMODE=1 并且 BIDIOE=1)

在此模式下，操作过程类似于只发送模式，不同的是：在使能 SPI 模块之前，需要在 SPI_CR2 寄存器中同时设置 BIDIMODE 和 BIDIOE 位为'1'。

单向只接收模式(BIDIMODE=0 并且 RXONLY=1)

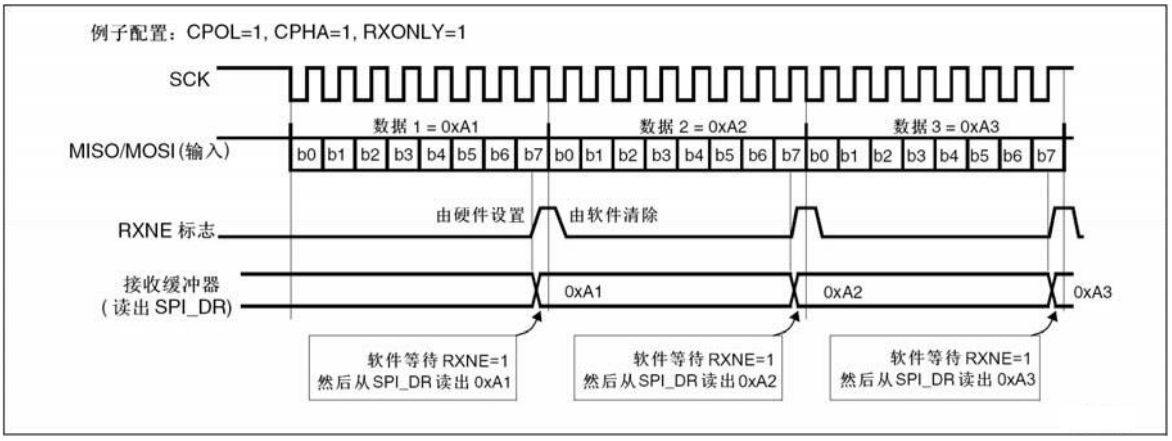
在此模式下，传输过程可以简要说明如下：

1. 在 SPI_CR2 寄存器中，设置 RXONLY=1；
2. 设置 SPE=1，使能 SPI 模块：
   a) 主模式下，立刻产生 SCK 时钟信号，在关闭 SPI(SPE=0)之前，不断地接收串行数据；
   b) 从模式下，当 SPI 主设备拉低 NSS 信号并产生 SCK 时钟时，接收串行数据。
3. 等待 RXNE=1，然后读出 SPI_DR 寄存器以获得收到的数据(同时会清除 RXNE 位)。重复这个操作接收所有数据。

也可以在响应 RXNE 标志的上升沿产生的中断的处理程序中实现这个过程。

注：　如果在最后一个数据传输结束后关闭 SPI 模块，请按照第 20.3.8 节的建议操作。

图 162 只接收模式(BIDIMODE=0 并且 RXONLY=1)下连续传输时，RXNE 变化示意图



单向接收过程(BIDIMODE=1 并且 BIDIOE=0)

在此模式下，操作过程类似于只接收模式，不同的是：在使能 SPI 模块之前，需要在 SPI_CR2 寄存器中设置 BIDIMODE 为'1'并清除 BIDIOE 位为'0'。

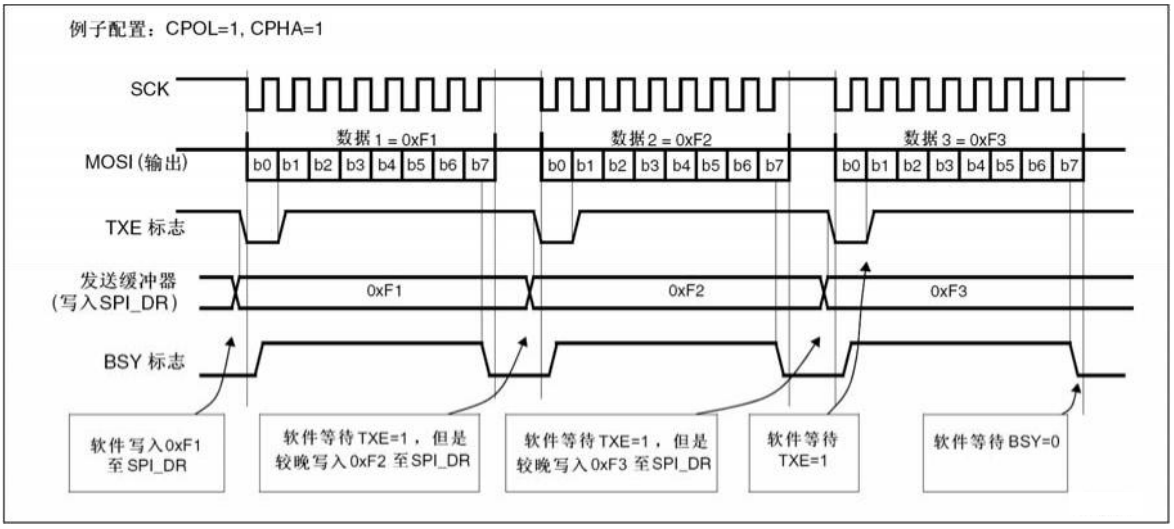
连续和非连续传输

当在主模式下发送数据时，如果软件足够快，能够在检测到每次 TXE 的上升沿(或 TXE 中断)，并立即在正在进行的传输结束之前写入 SPI_DR 寄存器，则能够实现连续的通信；此时，在每个数据项的传输之间的 SPI 时钟保持连续，同时 BSY 位不会被清除。

如果软件不够快，则会导致不连续的通信；这时，在每个数据传输之间会被清除(见下图)。 在主模式的只接收模式下(RXONLY=1)，通信总是连续的，而且 BSY 标志始终为'1'。

在从模式下，通信的连续性由 SPI 主设备决定。不管怎样，即使通信是连续的，BSY 标志会在每个数据项之间至少有一个 SPI 时钟周期为低(见图 161)。

图 163 非连续传输发送(BIDIMODE=0 并且 RXONLY=0)时，TXE/BSY 变化示意图



19.3.6 CRC 计算

CRC 校验用于保证全双工通信的可靠性。数据发送和数据接收分别使用单独的 CRC 计算器。通过对每一个接收位进行可编程的多项式运算来计算 CRC。CRC 的计算是在由 SPI_CR1 寄存器中CPHA 和 CPOL 位定义的采样时钟边沿进行的。

注意： *该 SPI 接口提供了两种 CRC 计算方法，取决于所选的发送和/或接收的数据帧格式：8 位数据帧采用 CR8；16 位数据帧采用 CRC16。*

CRC 计算是通过设置 SPI_CR1 寄存器中的 CRCEN 位启用的。设置 CRCEN 位时同时复位 CRC 寄存器(SPI_RXCRCR 和 SPI_TXCRCR)。在全双工和单发送模式下，如果传输是由软件控制（CPU 模式），需要在最后一个数据被写入 DR 寄存器后立即写 1 置位 CRCNEXT。在最后一个数据传输完成后，SPI_TXCRCR 的值会被发出。

在单接收模式下，如果传输是由软件控制（CPU 模式），需要在接收完倒数第二个数据的时候置位 CRCNEXT 位。CRC 数据会在最后一个数据之后接收，并且进行 CRC 结果比对。

在数据和 CRC 传输完成后，如果接收到的 CRC 数据与 SPI_RXCRCR 的内容不匹配，则 SPI_SR 寄存器的 CRCERR 标志位会被硬件置 1。

如果在 TX 缓冲器中还有数据，CRC 的数值仅在数据字节传输结束后传送。在传输 CRC 期间，CRC 计算器关闭，寄存器的数值保持不变。

SPI 通信可以通过以下步骤使用 CRC：

1. 设置 CPOL、CPHA、LSBFirst、BR、SSM、SSI 和 MSTR 的值；
2. 在 SPI_CRCPR 寄存器输入多项式；
3. 通过设置 SPI_CR1 寄存器 CRCEN 位使能 CRC 计算，该操作也会清除寄存器 SPI_RXCRCR 和 SPI_TXCRC；
4. 设置 SPI_CR1 寄存器的 SPE 位启动 SPI 功能；
5. 启动通信并且维持通信，直到只剩最后一个字节或者半字；
   - 在全双工和单发送模式下，如果是软件控制传输，那么需要在传输最后一个字节或者半字时通过置位 SPI_CR1 中的 CRCNEXT 位来指示在最后一个数据传输完成后会传输 CRC 值；
   - 在单接收模式下，需要在接收完倒数第二个数据开始接收最后一个数据的时候就置位 CRCNEXT 位，让 SPI 在接收完最后一个数据后进入 CRC 阶段。在 CRC 阶段的时候内部 CRC 计算会暂停。
6. 当最后一个字节或半字被发送后，SPI 进入 CRC 发送和校验阶段。在全双工和单接收模式下，接收到 的 CRC 与 SPI_RXCRCR 值进行比较。如果比较结果不等，则 SPI_SR 上的 CRCERR 标志位被置起。如果设置了 SPI_CR2 寄存器的 ERRIE 时，则同时会产生中断。

注意： *当 SPI 模块处于从设备模式时，请注意在时钟稳定之后再使能 CRC 计算，否则可能会得到错误的 CRC 计算结果。事实上，只要设置了 CRCEN 位，只要在 SCK 引脚上有输入时钟，不管 SPE 位的状态，都会进行 CRC 的计算。*

当 SPI 时钟频率较高时，用户在发送 CRC 时必须小心。在 CRC 传输期间，使用 CPU 的时间应尽可能少；为了避免在接收最后的数据和 CRC 时出错，在发送 CRC 过程中应禁止函数调用。必须在发送/接收最后一个数据之前完成设置 CRCNEXT 位的操作。

当 SPI 时钟频率较高时，因为 CPU 的操作会影响 SPI 的带宽，建议采用 DMA 模式以避免 SPI 降低的速度。

当 SPI 被配置为从模式并且使用了 NSS 硬件模式，NSS 引脚应该在数据传输和 CRC 传输期间保持为低。

当配置 SPI 为从模式并且使用 CRC 的功能，即使 NSS 信号为高时，如果 SCK 引脚上有时钟脉冲，则 CRC 计算会继续执行。例如在多从机的环境下，当主设备交替地与不同的从设备进行通信时，就会出现这种情况。

在不选中一个从设备(NSS 信号为高)转换到选中一个新的从设备(NSS 信号为低)的时候，为了保 持主从设备端下次 CRC 计算结果的同步，应该清除主从两端的 CRC 数值。

按照下述步骤清除 CRC 数值：

1. 关闭 SPI 模块(SPE=0);
2. 清除 CRCEN 位为’0’；
3. 设置 CRCEN 位为’1’；
4. 使能 SPI 模块(SPE=1)。

19.3.7 状态标志

应用程序通过 3 个状态标志可以完全监控 SPI 总线的状态。

发送缓冲器空闲标志(TXE)

此标志为'1'时表明发送缓冲器为空，可以写下一个待发送的数据进入缓冲器中。当写入 SPI_DR 时，TXE 标志被清除。

接收缓冲器非空(RXNE)

此标志为'1'时表明在接收缓冲器中包含有效的接收数据。读 SPI 数据寄存器可以清除此标志。

忙(Busy)标志

BSY 标志由硬件设置与清除(写入此位无效果)，此标志表明 SPI 通信层的状态。

当它被设置为'1'时，表明 SPI 正忙于通信，但有一个例外：在主模式的双向接收模式下(MSTR=1、BDM=1 并且 BDOE=0)，在接收期间 BSY 标志保持为低。

在软件要关闭 SPI 模块并进入停机模式(或关闭设备时钟)之前，可以使用 BSY 标志检测传输是否结束，这样可以避免破坏最后一次传输，因此需要严格按照下述过程执行。

BSY 标志还可以用于在多主系统中避免写冲突。

除了主模式的双向接收模式(MSTR=1、BDM=1 并且 BDOE=0)，当传输开始时，BSY 标志被置'1'。

以下情况时此标志将被清除为'0'：

- 当传输结束(主模式下，如果是连续通信的情况例外)；
- 当关闭 SPI 模块；
- 当产生主模式失效(MODF=1)。 如果通信不是连续的，则在每个

数据项的传输之间，BSY 标志为低。 当通信是连续时：

- 主模式下：在整个传输过程中，BSY 标志保持为高；
- 从模式下：在每个数据项的传输之间，BSY 标志在一个 SPI 时钟周期中为低。

注：　不要使用 BSY 标志处理每一个数据项的发送和接收，最好使用 TXE 和 RXNE 标志。

19.3.8 关闭 SPI

当通讯结束，可以通过关闭 SPI 模块来终止通讯。清除 SPE 位即可关闭 SPI。

在某些配置下，如果再传输还未完成时，就关闭 SPI 模块并进入停机模式，则可能导致当前的传输被破坏，而且 BSY 标志也变得不可信。

为了避免发生这种情况，关闭 SPI 模块时，建议按照下述步骤操作：

在主或从模式下的全双工模式(BIDIMODE=0，RXONLY=0)

1. 等待 RXNE=1 并接收最后一个数据；
2. 等待 TXE=1；
3. 等待 BSY=0；
4. 关闭 SPI(SPE=0)，最后进入停机模式(或关闭该模块的时钟)。

在主或从模式下的单向只发送模式(BIDIMODE=0，RXONLY=0)或双向的发送模式(BIDIMODE=1，BIDIOE=1)

在 SPI_DR 寄存器中写入最后一个数据后：

1. 等待 TXE=1；
2. 等待 BSY=0；
3. 关闭 SPI(SPE=0)，最后进入停机模式(或关闭该模块的时钟)。

在主或从模式下的单向只接收模式(MSTR=1，BIDIMODE=0，RXONLY=1)或双向 的接收模式(MSTR=1，BIDIMODE=1，BIDIOE=0)

这种情况需要特别地处理，以保证 SPI 不会开始一次新的传输：

1. 等待倒数第二个(第 n-1 个)RXNE=1；
2. 在关闭 SPI(SPE=0)之前等待一个 SPI 时钟周期(使用软件延迟)；
3. 在进入停机模式(或关闭该模块的时钟)之前等待最后一个 RXNE=1。

注：在主模式下的单向只发送模式(MSTR=1，BDM=1，BDOE=0)时，传输过程中 BSY 标志始终为低。

在从模式下的只接收模式(MSTR=0，BIDIMODE=0，RXONLY=1)或双向的接收模式(MSTR=0，BIDIMODE=1，BIDIOE=0)

1. 可以在任何时候关闭 SPI(SPE=0)，SPI 会在当前的传输结束后被关闭；
2. 如果希望进入停机模式，在进入停机模式(或关闭该模块的时钟)之前必须首先等待 BSY=0。

19.3.9 利用 DMA 的 SPI 通信

为了达到最大通信速度，需要及时往 SPI 发送缓冲器填数据，同样接收缓冲器中的数据也必须及时读走以防止溢出。为了方便高速率的数据传输，SPI 实现了一种采用简单的请求/应答的 DMA 机制。

当 SPI_CR2 寄存器上的对应使能位被设置时，SPI 模块可以发出 DMA 传输请求。发送缓冲器和接收缓冲器亦有各自的 DMA 请求。

- 发送时，在每次 TXE 被设置为'1'时发出 DMA 请求，DMA 控制器则写数据至 SPI_DR 寄存器，TXE 标志因此而被清除。
- 接收时，在每次 RXNE 被设置为' 1' 时发出 DMA 请求，DMA 控制器则从 SPI_DR 寄存器读出数据，RXNE 标志因此而被清除。

当只使用 SPI 发送数据时，只需使能 SPI 的发送 DMA 通道。此时，因为没有读取收到的数据，OVR 被置为'1'(译注：软件不必理会这个标志)。 当只使用 SPI 接收数据时，只需使能 SPI 的接收 DMA 通道。

在发送模式下，当 DMA 已经传输了所有要发送的数据(DMA_ISR 寄存器的 TCIF 标志变为'1')后，可以通过监视 BSY 标志以确认 SPI 通信结束，这样可以避免在关闭 SPI 或进入停止模式时，破坏最后一个数据的传输。因此软件需要先等待 TXE=1，然后等待 BSY=0。

注：在不连续的通信中，在写数据到 SPI_DR 的操作与 BSY 位被置为'1'之间，有 2 个 APB 时钟周期的延迟，因此，在写完最后一个数据后需要先等待 TXE=1 再等待 BSY=0。

图 164 使用 DMA 发送

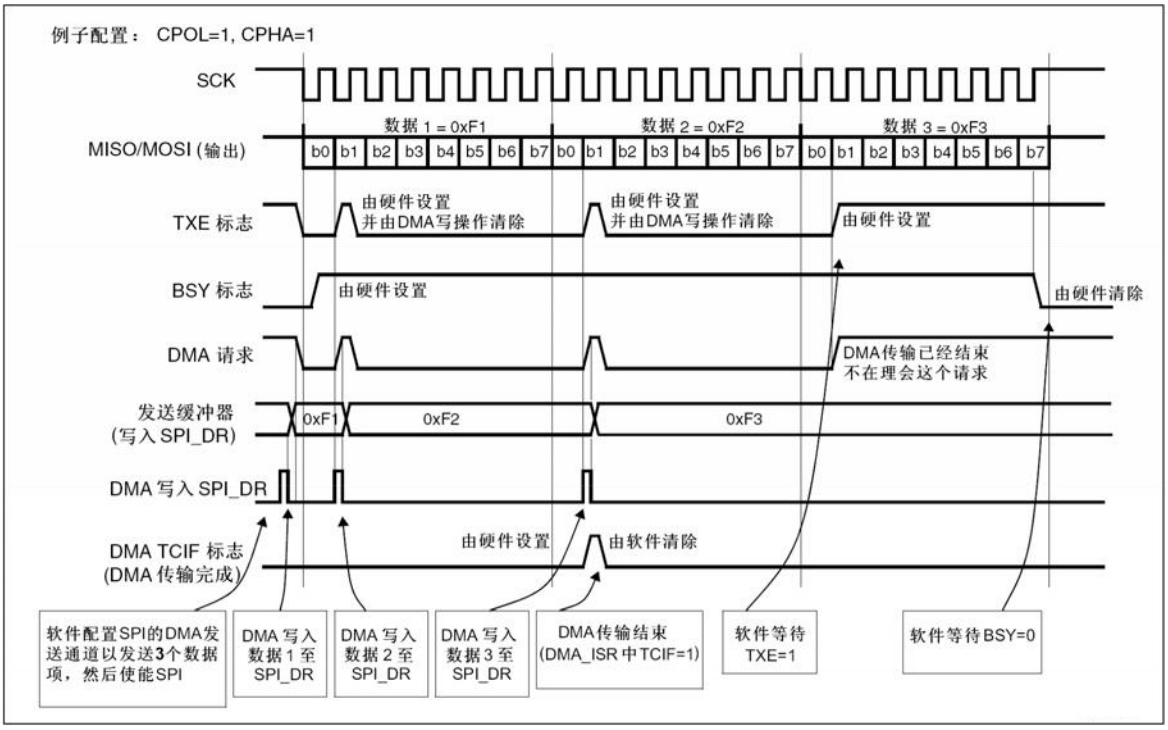
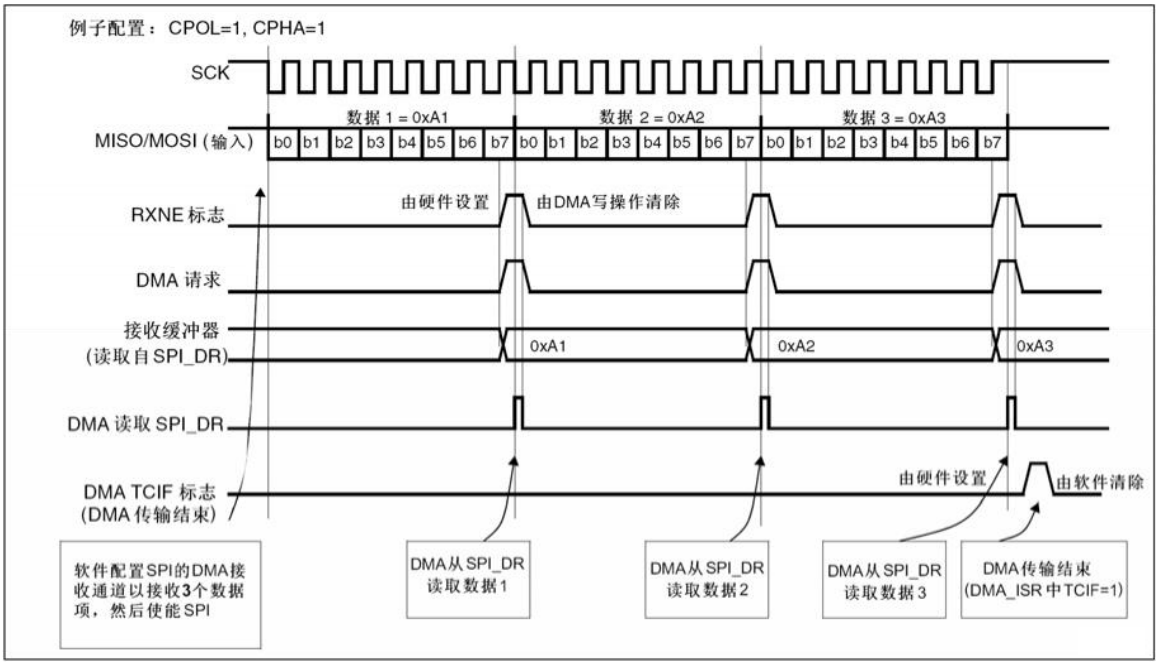


图 165 使用 DMA 接收



带 CRC 的 DMA 功能

当使能 SPI 使用 CRC 检验并且启用 DMA 模式时，在通信结束时，CRC 字节的发送和接收是自动完成的。

数据和 CRC 传输结束时，SPI_SR 寄存器的 CRCERR 标志为'1'表示在传输期间发生错误。

19.3.10 错误标志

主模式失效错误(MODF)

主模式失效仅发生在：NSS 引脚硬件模式管理下，主设备的 NSS 脚被拉低；或者在 NSS 引脚软件模式管理下，SSI 位被置为'0'时；MODF 位被自动置位。主模式失效对 SPI 设备有以下影响：

- MODF 位被置为'1'，如果设置了 ERRIE 位，则产生 SPI 中断；
- SPE 位被清为'0'。这将停止一切输出，并且关闭 SPI 接口；
- MSTR 位被清为'0'，因此强迫此设备进入从模式。

下面的步骤用于清除 MODF 位：

1. 当 MODF 位被置为'1'时，执行一次对 SPI_SR 寄存器的读或写操作；
2. 然后写 SPI_CR1 寄存器。

在有多个 MCU 的系统中，为了避免出现多个从设备的冲突，必须先拉高该主设备的 NSS 脚，再 对 MODF 位进行清零。在完成清零之后，SPE 和 MSTR 位可以恢复到它们的原始状态。

出于安全的考虑，当 MODF 位为'1'时，硬件不允许设置 SPE 和 MSTR 位。通常配置下，从设备的 MODF 位不能被置为'1'。然而，在多主配置里，一个设备可以在设置了 MODF 位的情况下，处于从设备模式；此时，MODF 位表示可能出现了多主冲突。中断程序可以执行一个复位或返回到默认状态来从错误状态中恢复。

溢出错误

当主设备已经发送了数据字节，而从设备还没有清除前一个数据字节产生的 RXNE 时，即为溢出错误。当产生溢出错误时：

- OVR 位被置为'1'；当设置了 ERRIE 位时，则产生中断。

此时，接收器缓冲器的数据不是主设备发送的新数据，读 SPI_DR 寄存器返回的是之前未读的数据，所有随后传送的数据都被丢弃。

依次读出 SPI_DR 寄存器和 SPI_SR 寄存器可将 OVR 清除。

CRC 错误

当设置了 SPI_CR 寄存器上的 CRCEN 位时，CRC 错误标志用来核对接收数据的有效性。如果移位寄存器中接收到的值(发送方发送的 SPI_TXCRCR 数值)与接收方 SPI_RXCRCR 寄存器中的数值不匹配，则 SPI_SR 寄存器上的 CRCERR 标志被置位为'1'。

19.3.11 SPI 中断

表 76 SPI 中断请求

中断事件	事件标志	使能控制位
发送缓冲器空标志	TXE	TXEIE
接收缓冲器非空标志	RXNE	RXNEIE
主模式失效事件	MODF	ERRIE
溢出错误	OVR	
CRC 错误标志	CRCERR	

19.4 SPI 寄存器描述

关于寄存器描述中所用到的缩略词可参见第 2.1 节。 可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

19.4.1 SPI 控制寄存器 1(SPI_CR1)

地址偏移：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRCEN	CRC NEXT	DFF	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15	**BIDIMODE**：双向数据模式使能(Bidirectional data mode enable) 0：选择“双线双向”模式； 1：选择“单线双向”模式。
位 14	**BIDIOE**：双向模式下的输出使能(Output enable in bidirectional mode) 和 BIDIMODE 位一起决定在“单线双向”模式下数据的输出方向 0：输出禁止(只收模式)； 1：输出使能(只发模式)。 这个“单线”数据线在主设备端为 MOSI 引脚，在从设备端为 MISO 引脚。
位 12	**CRCNEXT**：下一个发送 CRC (Transmit CRC next) 0：下一个发送的值来自发送缓冲区。 1：下一个发送的值来自发送 CRC 寄存器。 注：在 SPI_DR 寄存器写入最后一个数据后应马上设置该位。
位 11	**DFF**：数据帧格式(Data frame format) 0：使用 8 位数据帧格式进行发送/接收； 1：使用 16 位数据帧格式进行发送/接收。 注：只有当 SPI 禁止(SPE=0)时，才能写该位，否则出错。
位 10	**RXONLY**：只接收(Receive only) 该位和 BIDIMODE 位一起决定在“双线双向”模式下的传输方向。在多个从设备的配置中，在未被访问的从设备上该位被置 1，使得只有被访问的从设备有输出，从而不会造成数据线上数据冲突。 0：全双工(发送和接收)； 1：禁止输出(只接收模式)。
位 9	**SSM**：软件从设备管理(Software slave management) 当 SSM 被置位时，NSS 引脚上的电平由 SSI 位的值决定。 0：禁止软件从设备管理； 1：启用软件从设备管理。

位 8	**SSI**：内部从设备选择(Internal slave select) 该位只在 SSM 位为’1’时有意义。它决定了 NSS 上的电平，在 NSS 引脚上的 I/O 操作无效。
位 7	**LSBFIRST**：帧格式(Frame format) 0：先发送 MSB； 1：先发送 LSB。 注：当通信在进行时不能改变该位的值。
位 6	**SPE**：SPI 使能(SPI enable) 0：禁止 SPI 设备； 1：开启 SPI 设备。 注：当关闭 SPI 设备时，请按照第 20.3.8 节的过程操作。
位 5:3	**BR[2:0]**：波特率控制(Baud rate control) 000：$f_{PCLK}/2$　001：$f_{PCLK}/4$　010：$f_{PCLK}/8$　011：$f_{PCLK}/16$ 100：$f_{PCLK}/32$　101：$f_{PCLK}/64$　110：$f_{PCLK}/128$　111：$f_{PCLK}/256$ 当通信正在进行的时候，不能修改这些位。
位 2	**MSTR**：主设备选择(Master selection) 0：配置为从设备； 1：配置为主设备。 注：当通信正在进行的时候，不能修改该位。
位 1	**CPOL**：时钟极性(Clock polarity) 0：空闲状态时，SCK 保持低电平； 1：空闲状态时，SCK 保持高电平。 注：当通信正在进行的时候，不能修改该位。
位 0	**CPHA**：时钟相位(Clock phase) 0：数据采样从第一个时钟边沿开始； 1：数据采样从第二个时钟边沿开始。 注：当通信正在进行的时候，不能修改该位。

19.4.2 SPI 控制寄存器 2(SPI_CR2)

地址偏移：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								TXEIE	RXNEIE	ERRIE	保留		SSOE	TXDMAEN	RXDMAEN
res								rw	rw	rw	res		rw	rw	rw

位 15:8	保留位，硬件强制为 0
位 7	**TXEIE**：发送缓冲区空中断使能(Tx buffer empty interrupt enable) 0：禁止 TXE 中断； 1：允许 TXE 中断，当 TXE 标志置位为’1’时产生中断请求。
位 6	**RXNEIE**：接收缓冲区非空中断使能(RX buffer not empty interrupt enable) 0：禁止 RXNE 中断； 1：允许 RXNE 中断，当 RXNE 标志置位时产生中断请求。

位 5	**ERRIR**：错误中断使能(Error interrupt enable) 错误(CRCERR、OVR、MODF)产生时，该位控制是否产生中断 0： 禁止错误中断； 1：允许错误中断。
位 4:3	保留位，硬件强制为 0。
位 2	**SSOE**：SS 输出使能(SS output enable) 0：禁止在主模式下 SS 输出，该设备可以工作在多主设备模式； 1：设备开启时，开启主模式下 SS 输出，该设备不能工作在多主设备模式。
位 1	**TXDMAEN**：发送缓冲区 DMA 使能(Tx buffer DMA enable) 当该位被设置时，TXE 标志一旦被置位就发出 DMA 请求 0：禁止发送缓冲区 DMA； 1：启动发送缓冲区 DMA。
位 0	**RXDMAEN**：接收缓冲区 DMA 使能(Rx buffer DMA enable) 当该位被设置时，RXNE 标志一旦被置位就发出 DMA 请求 0：禁止接收缓冲区 DMA； 1：启动接收缓冲区 DMA。

19.4.3 SPI 状态寄存器(SPI_SR)

地址偏移：0x08

复位值：0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								BSY	OVR	MODF	CRC ERR	UDR	CHSIDE	TXE	RXNE
res								r	r	r	rc w0	r	r	r	r

位 15:8	保留位，硬件强制为 0
位 7	**BSY**：忙标志(Busy flag) 0：SPI 不忙； 1：SPI 正忙于通信，或者发送缓冲非空。 该位由硬件置位或者复位。 注：使用这个标志时需要特别注意，详见第 20.3.7 节和第 20.3.8 节。
位 6	**OVR**：溢出标志(Overrun flag) 0：没有出现溢出错误； 1：出现溢出错误。 该位由硬件置位，由软件序列复位。关于软件序列的详细信息，参考 20.4.7 节。
位 5	**MODF**：模式错误(Mode fault) 0：没有出现模式错误； 1：出现模式错误。 该位由硬件置位，由软件序列复位。关于软件序列的详细信息，参考 20.3.10 节。
位 4	**CRCERR**：CRC 错误标志(CRC error flag) 0：收到的 CRC 值和 SPI_RXCRCR 寄存器中的值匹配； 1：收到的 CRC 值和 SPI_RXCRCR 寄存器中的值不匹配。 该位由硬件置位，由软件写'0'而复位。

位 3	**UDR**：下溢标志位(Underrun flag) 0：未发生下溢； 1：发生下溢。 该标志位由硬件置'1'，由一个软件序列清'0'，详见 20.3.10 节。
位 2	**CHSIDE**：声道(Channel side) 0：需要传输或者接收左声道； 1：需要传输或者接收右声道。 注：在 SPI 模式下不使用。在 PCM 模式下无意义。
位 1	**TXE**：发送缓冲为空(Transmit buffer empty) 0：发送缓冲非空； 1：发送缓冲为空。
位 0	**RXNE**：接收缓冲非空(Receive buffer not empty) 0：接收缓冲为空； 1：接收缓冲非空。

19.4.4 SPI 数据寄存器(SPI_DR)

地址偏移：0x0C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0	**DR[15:0]**：数据寄存器(Data register) 待发送或者已经收到的数据 数据寄存器对应两个缓冲区：一个用于写(发送缓冲)；另外一个用于读(接收缓冲)。写操作将数据写到发送缓冲区；读操作将返回接收缓冲区里的数据。 **对 SPI 模式的注释：** 根据 SPI_CR1 的 DFF 位对数据帧格式的选择，数据的发送和接收可以是 8 位 或者 16 位的。为保证正确的操作，需要在启用 SPI 之前就确定好数据帧格式。 对于 8 位的数据，缓冲器是 8 位的，发送和接收时只会用到 SPI_DR[7:0]。在接收时，SPI_DR[15:8]被强制为 0。 对于 16 位的数据，缓冲器是 16 位的，发送和接收时会用到整个数据寄存器，即 SPI_DR[15:0]。

19.4.5 SPI CRC 多项式寄存器(SPI_CRCPR)

地址偏移：0x10

复位值：0x0007

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCPOLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0	**CRCPOLY[15:0]**：CRC 多项式寄存器(CRC polynomial register) 该寄存器包含了 CRC 计算时用到的多项式。其复位值为 0x0007， 根据应用可以设置其他数值。

19.4.6 SPI Rx CRC 寄存器(SPI_RXCRCR)

地址偏移：0x14

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RxCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 15:0	**RXCRC[15:0]**：接收 CRC 寄存器 在启用 CRC 计算时，RXCRC[15:0]中包含了依据收到的字节计算的 CRC 数值。当在 SPI_CR1 的 CRCEN 位写入'1'时，该寄存器被复位。CRC 计算使用 SPI_CRCPR 中的多项式。 当数据帧格式被设置为 8 位时，仅低 8 位参与计算，并且按照 CRC8 的方法进行；当数据帧格式 为 16 位时，寄存器中的所有 16 位都参与计算，并且按照 CRC16 的标准。 注：当 BSY 标志为'1'时读该寄存器，将可能读到不正确的数值。

19.4.7 SPI Tx CRC 寄存器(SPI_TXCRCR)

地址偏移：0x18

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TxCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 15:0	**TxCRC[15:0]**：发送 CRC 寄存器 在启用 CRC 计算时，TXCRC[15:0]中包含了依据将要发送的字节计算的 CRC 数值。当在 SPI_CR1 中的 CRCEN 位写入'1'时，该寄存器被复位。CRC 计算使用 SPI_CRCPR 中的多项式。 当数据帧格式被设置为 8 位时，仅低 8 位参与计算，并且按照 CRC8 的方法进行；当数据帧格式为 16 位时，寄存器中的所有 16 个位都参与计算，并且按照 CRC16 的标准。 注：当 BSY 标志为'1'时读该寄存器，将可能读到不正确的数值。

19. 4. 8 SPI 控制寄存器 SPI_CR3

地址偏移：0x40

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								FlexLenEn	保留		Len[4:0]				
								rw			rw	rw	rw	rw	rw

位 15:8	保留，始终读为 0。
位 7	**FlexLenEn**： 0 ： 不使用 2~32 比特数据包长度可配功能。 1 ： 使用 2~32 比特数据包长度可配功能，数据位宽由 Len[4:0]定义。
位 6:5	保留
位 4：0	**LEN**： 00000 ： 禁止设置。 Others ： 数据包长为 Len[4:0] + 1 个比特；例如 Len[4:0] = 5'b00101，数据包长为 6 比特

注意：　当使能2~32比特数据包长度可配功能后，SPI的CRC功能将无法使用。

19.4.9 SPI 寄存器地址映象

表 77 SPI 寄存器列表及其复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	SPI_CR1	保留																BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXOnly	SSM	SSI	LSBFIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004h	SPI_CR2	保留																								TXEIE	RXNEIE	ERRIE	保留		SSOE	TXDMAE	RXDMAE
	复位值																									0	0	0			0	0	0
008h	SPI_SR	保留																												保留			
	复位值																									0	0	0	0		1	0	
00Ch	SPI_DR	保留																DR[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	SPI_CRCPR	保留																CRCPOLY[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
014h	SPI_RXCRCR	保留																RxCRC[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	SPI_TXCRCR	保留																TxCRC[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

20 I2C 接口

20.1 I2C 简介

I²C(芯片间)总线接口连接微控制器和串行 I²C 总线。它提供多主机功能，控制所有 I²C 总线特定的时序、协议、仲裁和定时。支持标准和快速两种模式，同时与 SMBus 2.0 兼容。

I²C 模块有多种用途，包括 CRC 码的生成和校验、SMBus(系统管理总线—System Management Bus)和 PMBus(电源管理总线—Power Management Bus)。

根据特定设备的需要，可以使用 DMA 以减轻 CPU 的负担。

20.2 I2C 主要特点

- 并行总线/I²C 总线协议转换器
- 多主机功能：该模块既可做主设备也可做从设备
- I²C 主设备功能
  - 产生时钟
  - 产生起始和停止信号
- I²C 从设备功能
  - 可编程的 I²C 地址检测
  - 可响应 2 个从地址的双地址能力
  - 停止位检测
- 产生和检测 7 位/10 位地址和广播呼叫
- 支持不同的通讯速度
  - 标准速度(高达 100 kHz)
  - 快速(高达 400 kHz)
- 状态标志：
  - 发送器/接收器模式标志
  - 字节发送结束标志
  - I²C 总线忙标志
- 错误标志
  - 主模式时的仲裁丢失
  - 地址/数据传输后的应答(ACK)错误
  - 检测到错位的起始或停止条件
  - 禁止拉长时钟功能时的上溢或下溢
- 2 个中断向量
  - 1 个中断用于地址/数据通讯成功
  - 1 个中断用于错误
- 可选的拉长时钟功能
- 具单字节缓冲器的 DMA
- 可配置的 PEC(信息包错误检测)的产生或校验：
  - 发送模式中 PEC 值可以作为最后一个字节传输
  - 用于最后一个接收字节的 PEC 错误校验
- 兼容 SMBus 2.0
  - 25 ms 时钟低超时延时

- 10 ms 主设备累积时钟低扩展时间
- 25 ms 从设备累积时钟低扩展时间
- 带 ACK 控制的硬件 PEC 产生/校验
- 支持地址分辨协议(ARP)

- 兼容 SMBus

注：不是所有产品中都包含上述所有特性。请参考相关的数据手册，确认该产品支持的 I^2C 功能。

20.3 I2C 功能描述

I^2C 模块接收和发送数据，并将数据从串行转换成并行，或并行转换成串行。可以开启或禁止中断。接口通过数据引脚(SDA)和时钟引脚(SCL)连接到 I^2C 总线。允许连接到标准(高达 100kHz)或快速(高达 400kHz)的 I^2C 总线。

20.3.1 模式选择

接口可以下述 4 种模式中的一种运行：

- 从发送器模式
- 从接收器模式
- 主发送器模式
- 主接收器模式

该模块默认地工作于从模式。接口在生成起始条件后自动地从从模式切换到主模式；当仲裁丢失或产生停止信号时，则从主模式切换到从模式。允许多主机功能。

通信流

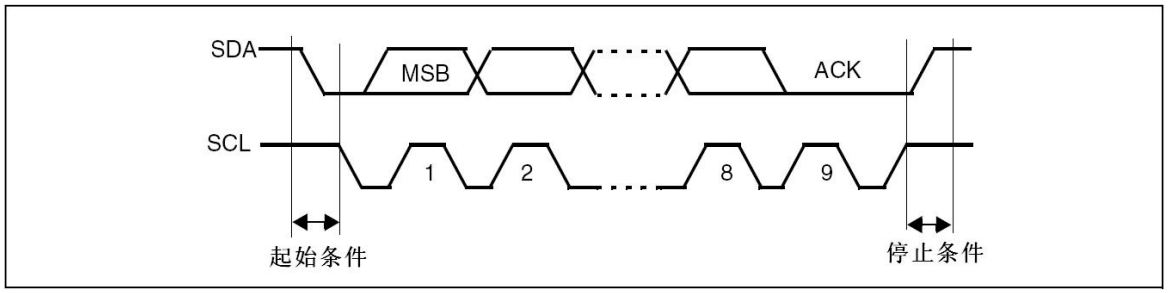
主模式时，I^2C 接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始并以停止条件结束。起始条件和停止条件都是在主模式下由软件控制产生。

从模式时，I^2C 接口能识别它自己的地址(7 位或 10 位)和广播呼叫地址。软件能够控制开启或禁止广播呼叫地址的识别。

数据和地址按 8 位/字节进行传输，高位在前。跟在起始条件后的 1 或 2 个字节是地址(7 位模式为 1 个字节，10 位模式为 2 个字节)。地址只在主模式发送。

在一个字节传输的 8 个时钟后的第 9 个时钟期间，接收器必须回送一个应答位(ACK)给发送器。参考下图。

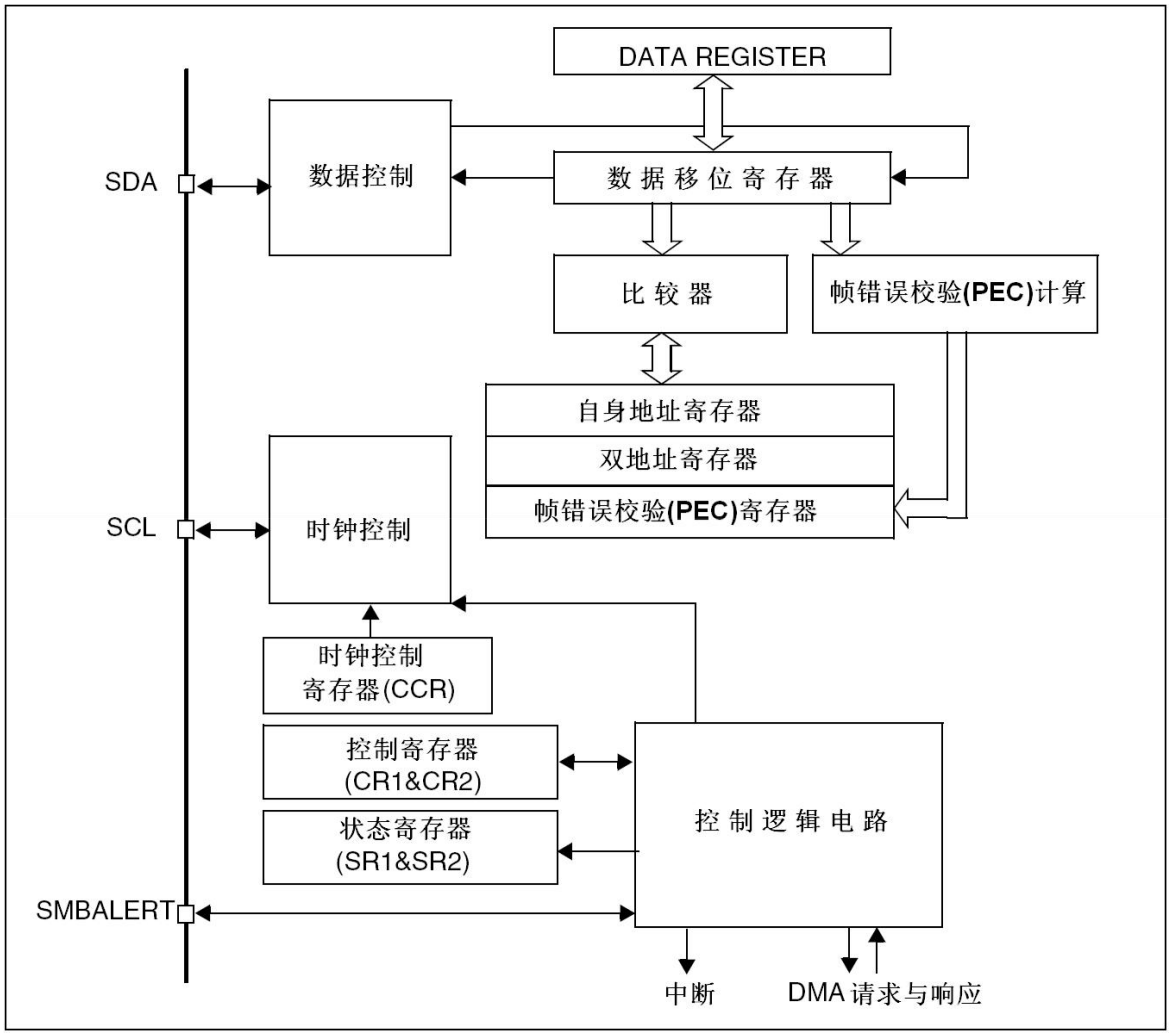
图 166 I2C 总线协议



软件可以开启或禁止应答(ACK)，并可以设置 I^2C 接口的地址(7 位、10 位地址或广播呼叫地址)。

I^2C 接口的功能框图示于下图。

图 167 I2C 的功能框图



注： *在 SMBus 模式下，SMBALERT 是可选信号。如果禁止了 SMBus，则不能使用该信号。*

20.3.2 I2C 从模式

默认情况下，I^2C 接口总是工作在从模式。从从模式切换到主模式，需要产生一个起始条件。

为了产生正确的时序，必须在 I2C_CR2 寄存器中设定该模块的输入时钟。输入时钟的频率必须至少是：

- 标准模式下为：2MHz
- 快速模式下为：4MHz

一旦检测到起始条件，在 SDA 线上接收到的地址被送到移位寄存器。然后与芯片自己的地址 OAR1 和 OAR2(当 ENDUAL=1)或者广播呼叫地址(如果 ENGC=1)相比较。

注： *在 10 位地址模式时，比较包括头段序列(11110xx0)，其中的 xx 是地址的两个最高有效位。*

头段或地址不匹配：I^2C 接口将其忽略并等待另一个起始条件。

头段匹配(仅 10 位模式)：如果 ACK 位被置'1'，I^2C 接口产生一个应答脉冲并等待 8 位从地址。

地址匹配：I^2C 接口产生以下时序：

- 如果 ACK 被置'1'，则产生一个应答脉冲
- 硬件设置 ADDR 位；如果设置了 ITEVFEN 位，则产生一个中断
- 如果 ENDUAL=1，软件必须读 DUALF 位，以确认响应了哪个从地址。

在 10 位模式，接收到地址序列后，从设备总是处于接收器模式。在收到与地址匹配的头序列并且最低位为'1'(即 11110xx1)后，当接收到重复的起始条件时，将进入发送器模式。

在从模式下 TRA 位指示当前是处于接收器模式还是发送器模式。

从发送器

在接收到地址和清除 ADDR 位后，从发送器将字节从 DR 寄存器经由内部移位寄存器发送到 SDA 线上。

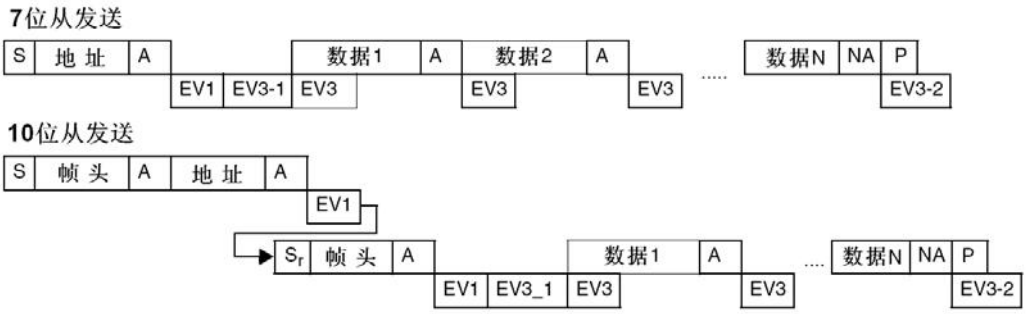
从设备保持 SCL 为低电平，直到 ADDR 位被清除并且待发送数据已写入 DR 寄存器。(见下图中的 EV1 和 EV3)。

当收到应答脉冲时:

- TxE 位被硬件置位，如果设置了 ITEVFEN 和 ITBUFEN 位，则产生一个中断。

如果 TxE 位被置位，但在下一个数据发送结束之前没有新数据写入到 I2C_DR 寄存器，则 BTF 位被置位，在清除 BTF 之前 I^2C 接口将保持 SCL 为低电平；读出 I2C_SR1 之后再写入 I2C_DR 寄存器 将清除 BTF 位。

图 168 从发送器的传送序列图



说明：S=Start(起始条件)，S_r=重复的起始条件，P=Stop(停止条件)，A=响应，NA=非响应，EVx=事件(ITEVFEN=1 时产生中断)

EV1：ADDR=1，读 SR1 然后读 SR2 将清除该事件。
EV3-1：TxE=1，移位寄存器空，数据寄存器空，写 DR。 EV3：TxE=1，移位寄存器非空，数据寄存器空，写 DR 将清除该事件。 EV3-2：AF=1，在 SR1 寄存器的 AF 位写'0'可清除 AF 位。

注：1 – EV1 和 EV3_1 事件拉长 SCL 低的时间，直到对应的软件序列结束。
2 – EV3 的软件序列必须在当前字节传输结束之前完成。

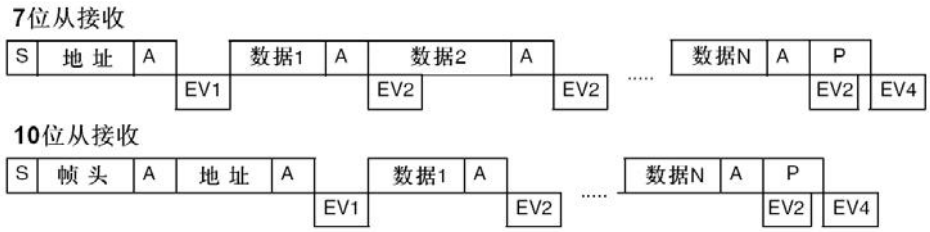
从接收器

在接收到地址并清除 ADDR 后，从接收器将通过内部移位寄存器从 SDA 线接收到的字节存进 DR 寄存器。I^2C 接口在接收到每个字节后都执行下列操作:

- 如果设置了 ACK 位，则产生一个应答脉冲
- 硬件设置 RxNE=1。如果设置了 ITEVFEN 和 ITBUFEN 位，则产生一个中断。

如果 RxNE 被置位，并且在接收新的数据结束之前 DR 寄存器未被读出，BTF 位被置位，在清除 BTF 之前 I^2C 接口将保持 SCL 为低电平；读出 I2C_SR1 之后再写入 I2C_DR 寄存器将清除 BTF 位。(见下图)。

图 169 从接收器的传送序列图



说明：S=Start(起始条件)，S_r=重复的起始条件，P=Stop(停止条件)，A=响应，NA=非响应，EVx=事件(ITEVFEN=1 时产生中断)

EV1：ADDR=1，读 SR1 然后读 SR2 将清除该事件。 EV2：RxNE=1，读 DR 将清除该事件。 EV4：STOPF=1，读 SR1 然后写 CR1 寄存器将清除该事件。

注：1 – EV1 事件拉长 SCL 低的时间，直到对应的软件序列结束。
2 – EV2 的软件序列必须在当前字节传输结束之前完成。

关闭从通信

在传输完最后一个数据字节后，主设备产生一个停止条件，I^2C 接口检测到这一条件时：

- 设置 STOPF=1，如果设置了 ITEVFEN 位，则产生一个中断。

然后 I^2C 接口等待读 SR1 寄存器，再写 CR1 寄存器。(见上图的 EV4)。

20.3.3 I2C 主模式

在主模式时，I^2C 接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始并以停 止条件结束。当通过 START 位在总线上产生了起始条件，设备就进入了主模式。

以下是主模式所要求的操作顺序：

- 在 I2C_CR2 寄存器中设定该模块的输入时钟以产生正确的时序
- 配置时钟控制寄存器
- 配置上升时间寄存器
- 编程 I2C_CR1 寄存器启动外设
- 置 I2C_CR1 寄存器中的 START 位为 1，产生起始条件

I^2C 模块的输入时钟频率必须至少是:

- 标准模式下为：2MHz
- 快速模式下为：4MHz

起始条件

当 BUSY=0 时，设置 START=1，I^2C 接口将产生一个开始条件并切换至主模式(M/SL 位置位)。

注：在主模式下，设置 START 位将在当前字节传输完后由硬件产生一个重开始条件。

一旦发出开始条件:

- SB 位被硬件置位，如果设置了 ITEVFEN 位，则会产生一个中断。

然后主设备等待读 SR1 寄存器，紧跟着将从地址写入 DR 寄存器(见图 170 和图 171 的 EV5)。

从地址的发送

从地址通过内部移位寄存器被送到 SDA 线上。

- 在 10 位地址模式时，发送一个头段序列产生以下事件：
  - ADD10 位被硬件置位，如果设置了 ITEVFEN 位，则产生一个中断。

然后主设备等待读 SR1 寄存器，再将第二个地址字节写入 DR 寄存器(见图 170 和图 171)。

– ADDR 位被硬件置位，如果设置了 ITEVFEN 位，则产生一个中断。

随后主设备等待一次读 SR1 寄存器，跟着读 SR2 寄存器(见图 170 和图 171)。

- 在 7 位地址模式时，只需送出一个地址字节。

一旦该地址字节被送出，

– ADDR 位被硬件置位，如果设置了 ITEVFEN 位，则产生一个中断。

随后主设备等待一次读 SR1 寄存器，跟着读 SR2 寄存器(见图 170 和图 171)。

根据送出从地址的最低位，主设备决定进入发送器模式还是进入接收器模式。

- 在 7 位地址模式时，
  – 要进入发送器模式，主设备发送从地址时置最低位为'0'。
  – 要进入接收器模式，主设备发送从地址时置最低位为'1'。
- 在 10 位地址模式时
  – 要进入发送器模式，主设备先送头字节(11110xx0)，然后送最低位为'0'的从地址。(这里 xx 代表 10 位地址中的最高 2 位。)
  – 要进入接收器模式，主设备先送头字节(11110xx0)，然后送最低位为'1'的从地址。然后再 重新发送一个开始条件，后面跟着头字节(11110xx1)(这里 xx 代表 10 位地址中的最高 2 位。)

TRA 位指示主设备是在接收器模式还是发送器模式。

主发送器

在发送了地址和清除了 ADDR 位后,主设备通过内部移位寄存器将字节从 DR 寄存器发送到 SDA线上。

主设备等待，直到 TxE 被清除，(见图 170 的 EV8)。

当收到应答脉冲时:

- TxE 位被硬件置位，如果设置了 INEVFEN 和 ITBUFEN 位，则产生一个中断。

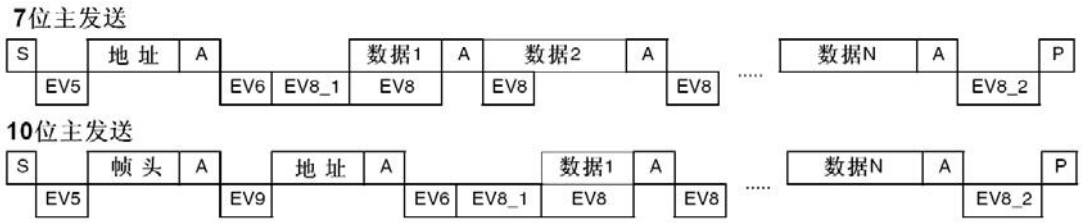
如果 TxE 被置位并且在上一次数据发送结束之前没有写新的数据字节到 DR 寄存器，则 BTF 被硬件置位，在清除 BTF 之前 I^2C 接口将保持 SCL 为低电平；读出 I2C_SR1 之后再写入 I2C_DR 寄存器将清除 BTF 位。

关闭通信

在 DR 寄存器中写入最后一个字节后，通过设置 STOP 位产生一个停止条件(见图 170 的 EV8_2)，然后 I^2C 接口将自动回到从模式(M/S 位清除)。

注： *当 TxE 或 BTF 位置位时，停止条件应安排在出现 EV8_2 事件时。*

图 170 主发送器传送序列图



说明：S=Start(起始条件)，S_r=重复的起始条件，P=Stop(停止条件)，A=响应，NA=非响应，EVx=事件(ITEVFEN=1 时产生中断)。

EV5：SB=1，读 SR1 然后将地址写入 DR 寄存器将清除该事件。 EV6：ADDR=1，读 SR1 然后读 SR2 将清除该事件。 EV8_1：TxE=1，移位寄存器空，数据寄存器空，写 DR 寄存器。EV8：TxE=1，移位寄存器非空，数据寄存器空，写入 DR 寄存器将清除该事件。EV8_2：TxE=1，BTF=1，请求设置停止位。TxE 和 BTF 位由硬件在产生停止条件时清除。 EV9：ADDR10=1，读 SR1 然后写入 DR 寄存器将清除该事件。

注：1 – EV5、EV6、EV9、EV8_1 和 EV8_2 事件拉长 SCL 低的时间，直到对应的软件序列结束。
2 – EV8 的软件序列必须在当前字节传输结束之前完成。

主接收器

在发送地址和清除 ADDR 之后，I^2C 接口进入主接收器模式。在此模式下，I^2C 接口从 SDA 线接收数据字节，并通过内部移位寄存器送至 DR 寄存器。在每个字节后，I^2C 接口依次执行以下操作：

- 如果 ACK 位被置位，发出一个应答脉冲。
- 硬件设置 RxNE=1，如果设置了 INEVFEN 和 ITBUFEN 位，则会产生一个中断(见图 171 的 EV7)。

如果 RxNE 位被置位，并且在接收新数据结束前，DR 寄存器中的数据没有被读走，硬件将设置BTF=1，在清除 BTF 之前 I^2C 接口将保持 SCL 为低电平；读出 I2C_SR1 之后再读出 I2C_DR 寄存器将清除 BTF 位。

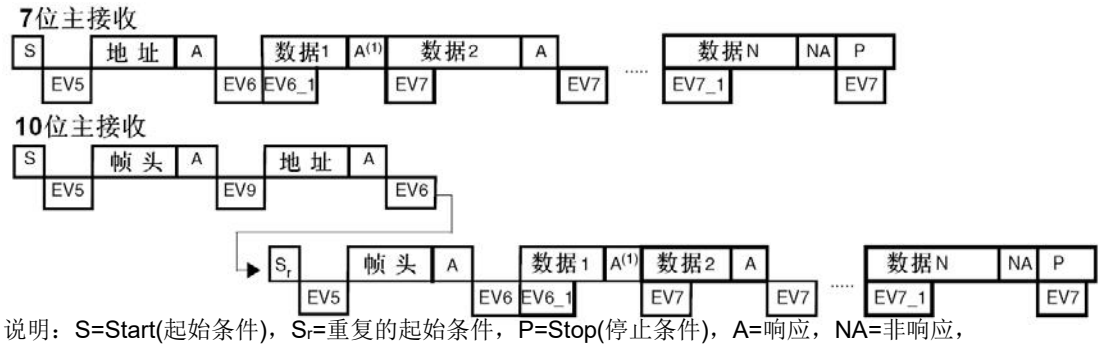
关闭通信

主设备在从从设备接收到最后一个字节后发送一个NACK。接收到NACK后，从设备释放对SCL 和 SDA 线的控制；主设备就可以发送一个停止/重起始条件。

- 为了在收到最后一个字节后产生一个 NACK 脉冲，在读倒数第二个数据字节之后(在倒数第二个 RxNE 事件之后)必须清除 ACK 位。
- 为了产生一个停止/重起始条件，软件必须在读倒数第二个数据字节之后(在倒数第二个 RxNE 事件之后)设置 STOP/START 位。
- 只接收一个字节时，刚好在 EV6 之后(EV6_1 时，清除 ADDR 之后)要关闭应答和停止条件的产生位。

在产生了停止条件后，I^2C 接口自动回到从模式(M/SL 位被清除)。

图 171 主接收器传送序列图



说明：S=Start(起始条件)，S_r=重复的起始条件，P=Stop(停止条件)，A=响应，NA=非响应，EVx=事件(ITEVFEN=1 时产生中断)

EV5：SB=1，读 SR1 然后将地址写入 DR 寄存器将清除该事件。
EV6：ADDR=1，读 SR1 然后读SR2 将清除该事件。在10位主接收模式下，该事件后应设置CR2 的
START=1。
EV6_1：没有对应的事件标志，只适于接收 1 个字节的情况。恰好在 EV6 之后(即清除了 ADDR 之后)，要清 除响应和停止条件的产生位。
EV7：RxNE=1，读 DR 寄存器清除该事件。 EV7_1：RxNE=1，读 DR 寄存器清除该事件。设置 ACK=0 和 STOP 请求。 EV9：ADDR10=1，读 SR1 然后写入 DR 寄存器将清除该事件。

1. 如果收到一个单独的字节，则是 NA。
2. EV5、EV6 和 EV9 事件拉长 SCL 低电平，直到对应的软件序列结束。
3. EV7 的软件序列必须在当前字节传输结束前完成。
4. EV6_1 或 EV7_1 的软件序列必须在当前传输字节的 ACK 脉冲之前完成。

20.3.4 错误条件

以下条件可能造成通讯失败。

总线错误(BERR)

在一个地址或数据字节传输期间，当 I^2C 接口检测到一个外部的停止或起始条件则产生总线错误。此时：

- BERR 位被置位为'1'；如果设置了 ITERREN 位，则产生一个中断；
- 在从模式情况下，数据被丢弃，硬件释放总线：
  - 如果是错误的开始条件，从设备认为是一个重启动，并等待地址或停止条件。
  - 如果是错误的停止条件，从设备按正常的停止条件操作，同时硬件释放总线。
- 在主模式情况下，硬件不释放总线，同时不影响当前的传输状态。此时由软件决定是否要 中止当前的传输。

应答错误(AF)

当接口检测到一个无应答位时，产生应答错误。此时：

- AF 位被置位，如果设置了 ITERREN 位，则产生一个中断；
- 当发送器接收到一个 NACK 时，必须复位通讯：
  - 如果是处于从模式，硬件释放总线。
  - 如果是处于主模式，软件必须生成一个停止条件。

仲裁丢失(ARLO)

当 I^2C 接口检测到仲裁丢失时产生仲裁丢失错误，此时：

- ARLO 位被硬件置位，如果设置了 ITERREN 位，则产生一个中断；
- I^2C 接口自动回到从模式(M/SL 位被清除)。当 I^2C 接口丢失了仲裁，则它无法在同一个传输中响应它的从地址，但它可以在赢得总线的主设备发送重起始条件之后响应；
- 硬件释放总线。

过载/欠载错误(OVR)

在从模式下，如果禁止时钟延长，I^2C 接口正在接收数据时，当它已经接收到一个字节(RxNE=1)，但在 DR 寄存器中前一个字节数据还没有被读出，则发生过载错误。此时：

- 最后接收的数据被丢弃；
- 在过载错误时，软件应清除 RxNE 位，发送器应该重新发送最后一次发送的字节。

在从模式下，如果禁止时钟延长，I^2C 接口正在发送数据时，在下一个字节的时钟到达之前，新的数据还未写入 DR 寄存器(TxE=1)，则发生欠载错误。此时：

- 在 DR 寄存器中的前一个字节将被重复发出；
- 用户应该确定在发生欠载错时，接收端应丢弃重复接收到的数据。发送端应按 I^2C 总线标准在规定的时间更新 DR 寄存器。

在发送第一个字节时，必须在清除 ADDR 之后并且第一个 SCL 上升沿之前写入 DR 寄存器；如果不能做到这点，则接收方应该丢弃第一个数据。

20.3.5 SDA/SCL 线控制

- 如果允许时钟延长：
  - 发送器模式：如果 TxE=1 且 BTF=1：I^2C 接口在传输前保持时钟线为低，以等待软件读取 SR1，然后把数据写进数据寄存器(缓冲器和移位寄存器都是空的)。
  - 接收器模式：如果 RxNE=1 且 BTF=1：I^2C 接口在接收到数据字节后保持时钟线为低，以 等待软件读 SR1，然后读数据寄存器 DR(缓冲器和移位寄存器都是满的)。
- 如果在从模式中禁止时钟延长：
  - 如果 RxNE=1，在接收到下个字节前 DR 还没有被读出，则发生过载错。接收到的最后一个字节丢失。
  - 如果 TxE=1，在必须发送下个字节之前却没有新数据写进 DR，则发生欠载错。相同的字节将被重复发出。
  - 不控制重复写冲突。

20.3.6 SMBus

介绍

系统管理总线(SMBus)是一个双线接口。通过它，各设备之间以及设备与系统的其他部分之间可以互相通信。它基于 I^2C 操作原理。SMBus 为系统和电源管理相关的任务提供一条控制总线。一个系统利用 SMBus 可以和多个设备互传信息，而不需使用独立的控制线路。

系统管理总线(SMBus)标准涉及三类设备。

从设备：接收或响应命令的设备。

主设备：用来发送命令、产生时钟和终止发送的设备。

主机：一种专用的主设备，它提供与系统 CPU 的主接口。主机必须具有主-从机功能并且必须支持 SMBus 提醒协议。一个系统里只允许有一个主机。

SMBus 和 I^2C 之间的相似点

- 2 条线的总线协议(1 个时钟，1 个数据) + 可选的 SMBus 提醒线；
- 主-从通信，主设备提供时钟；
- 多主机功能
- SMBus 数据格式类似于 I^2C 的 7 位地址格式(见图 166)；

SMBus 和 I2C 之间的不同点

下表列出了 SMBus 和 I^2C 的不同点。

表 78 SMBus 与 I2C 的比较

SMBus	I^2C
最大传输速度 100kHz	最大传输速度 400kHz
最小传输速度 10kHz	无最小传输速度
35ms 时钟低超时	无时钟超时
固定的逻辑电平	逻辑电平由 VDD 决定
不同的地址类型(保留的、动态的等)	7 位、10 位和广播呼叫从地址类型
不同的总线协议(快速命令、处理呼叫等)	无总线协议

SMBus 应用用途

利用系统管理总线，设备可提供制造商信息，告诉系统它的型号/部件号，保存暂停事件的状态，报告不同类型的错误，接收控制参数，和返回它的状态。SMBus 为系统和电源管理相关的任务提供控制总线。

设备标识

在系统管理总线上，任何一个作为从模式的设备都有一个唯一的地址，叫做从地址。保留的从地址表请参考 2.0 版的 SMBus 规范(*http://smbus.org/specs/*)。

总线协议

SMBus 技术规范支持 9 个总线协议。有关这些协议的详细资料和 SMBus 地址类型，请参考 2.0 版 的 SMBus 规范(*http://smbus.org/specs/*)。这些协议由用户的软件来执行。

地址解析协议(ARP)

通过给每个从设备动态地分配一个新的唯一地址，可以解决 SMBus 的从地址冲突。地址解析协议(ARP)具有以下的特性：

- 使用标准 SMBus 物理层仲裁机制分配地址；

- 当设备维持供电期间，分配的地址仍保持不变，也允许设备在断电后保留其地址。
- 在地址分配后，没有额外的 SMBus 的打包开销(也就是说访问分配地址的设备与访问固定地址的设备所用时间是一样的)；
- 任何一个 SMBus 主设备可以遍历总线。

唯一的设备标识符(UDID)

为了分配地址，需要一种区分每个设备的机制，每个设备必须拥有一个唯一的设备标识符。

关于在 ARP 上 128 位的 UDID 的详细信息，参考 2.0 版的 SMBus 规范(*http://smbus.org/specs/*)。

SMBus 提醒模式

SMBus 提醒是一个带中断线的可选信号，用于那些希望扩展它们的控制能力而牺牲一个引脚的设备。SMBALERT 和 SCL、SDA 信号一样，是一种线与信号。SMBALERT 通常和 SMBus 广播呼叫地址一起使用。与 SMBus 有关的消息为 2 字节。

一个只具有从功能的设备，可以通过设置 I2C_CR1 寄存器上的 ALERT 位，使用 SMBALERT 给主机发信号表示它希望进行通信。主机处理该中断并通过提醒响应地址 ARA(*Alert Response Address*，地址值为 0001100x)访问所有 SMBALERT 设备。只有那些将 SMBALERT 拉低的设备能应答 ARA。此状态是由 I2C_SR1 寄存器中的 SMBALERT 状态标记来标识的。主机执行一个修改过的接收字节操作。由从发送设备提供的 7 位设备地址被放在字节的 7 个最高位上，第八个位 可以是'0'或'1'。

如果多个设备把 SMBALERT 拉低，最高优先级设备(最小的地址)将在地址传输期间通过标准仲 裁赢得通信权。在确认从地址后，此设备不得再拉低它的 SMBALERT，如果当信息传输完成后，主机仍看到 SMBALERT 低，就知道需要再次读 ARA。

没有实现 SMBALERT 信号的主机可以定期访问 ARA。 有关 SMBus 提醒模式的更多详细资料，请参考 2.0 版的 SMBus 规范(*http://smbus.org/specs/*)。

超时错误

在定时规范上 I^2C 和 SMBus 之间有很多差别。

SMBus 定义了一个时钟低超时，35ms 的超时。SMBus 规定 TLOW:SEXT 为从设备的累积时钟低扩展时间。SMBus 规定 TLOW:MEXT 为主设备的累积时钟低扩展时间。更多超时细节请参考 2.0 版的 SMBus 规范(*http://smbus.org/specs/*)。

I2C_SR1 中的状态标志 Timeout 或 Tlow 错误表明了这个特性的状态。

如何使用 SMBus 模式的接口

为了从 I^2C 模式切换到 SMBus 模式，应该执行下列步骤：

- 设置 I2C_CR1 寄存器中的 SMBus 位；
- 按应用要求配置 I2C_CR1 寄存器中的 SMBTYPE 和 ENARP 位。如果要把设备配置成主设备，产生起始条件的步骤见 21.3.3 节 I2C 主模式。否则，参见 21.3.2 节 I2C 从模式。

软件程序必须处理多种 SMBus 协议。

- 如果 ENARP=1 且 SMBTYPE=0，使用 SMB 设备默认地址。
- 如果 ENARP=1 且 SMBTYPE=1，使用 SMB 主设备头字段。
- 如果 SMBALERT=1，使用 SMB 提醒响应地址。

20.3.7 DMA 请求

DMA 请求(当被使能时)仅用于数据传输。发送时数据寄存器变空或接收时数据寄存器变满，则产生 DMA 请求。DMA 请求必须在当前字节传输结束之前被响应。当为相应 DMA 通道设置的数据传输量已经完成时，DMA 控制器发送传输结束信号 ETO 到 I^2C 接口，并且在中断允许时产生一个传输完成中断：

- 主发送器：在 EOT 中断服务程序中，需禁止 DMA 请求，然后在等到 BTF 事件后设置停止条件。
- 主接收器：当要接收的数据数目大于或等于 2 时，DMA 控制器发送一个硬件信号 EOT_1，它 对应 DMA 传输(字节数－1)。如果在 I2C_CR2 寄存器中设置了 LAST 位，硬件在发送完 EOT_1 后的下一个字节，将自动发送 NACK。在中断允许的情况下，用户可以在 DMA 传输 完成的中断服务程序中产生一个停止条件。

利用 DMA 发送

通过设置 I2C_CR2 寄存器中的 DMAEN 位可以激活 DMA 模式。只要 TxE 位被置位，数据将由DMA 从预置的存储区装载进 I2C_DR 寄存器。为 I^2C 分配一个 DMA 通道，须执行以下步骤(x 是通道号)：

1. 在 DMA_CPARx 寄存器中设置 I2C_DR 寄存器地址。数据将在每个 TxE 事件后从存储器传 送至这个地址。
2. 在 DMA_CMARx 寄存器中设置存储器地址。数据在每个 TxE 事件后从这个存储区传送至 I2C_DR。
3. 在 DMA_CNDTRx 寄存器中设置所需的传输字节数。在每个 TxE 事件后，此值将被递减。
4. 利用 DMA_CCRx 寄存器中的 PL[0:1]位配置通道优先级。
5. 设置 DMA_CCRx 寄存器中的 DIR 位，并根据应用要求可以配置在整个传输完成一半或全部完成时发出中断请求。
6. 通过设置 DMA_CCTx 寄存器上的 EN 位激活通道。 当 DMA 控制器中设置的数据传输数目已经完成时，DMA 控制器给 I^2C 接口发送一个传输结束的EOT/ EOT_1 信号。在中断允许的情况下，将产生一个 DMA 中断。

注： 如果使用 DMA 进行发送时，不要设置 I2C_CR2 寄存器的 ITBUFEN 位。

利用 DMA 接收

通过设置 I2C_CR2 寄存器中的 DMAEN 位可以激活 DMA 接收模式。每次接收到数据字节时，将由 DMA 把 I2C_DR 寄存器的数据传送到设置的存储区(参考 DMA 说明)。设置 DMA 通道进行 I^2C 接收，须执行以下步骤(x 是通道号)：

1. 在 DMA_CPARx 寄存器中设置 I2C_DR 寄存器的地址。数据将在每次 RxNE 事件后从此地址传送到存储区。
2. 在 DMA_CMARx 寄存器中设置存储区地址。数据将在每次 RxNE 事件后从 I2C_DR 寄存器 传送到此存储区。
3. 在 DMA_CNDTRx 寄存器中设置所需的传输字节数。在每个 RxNE 事件后，此值将被递减。
4. 用 DMA_CCRx 寄存器中的 PL[0:1]配置通道优先级。
5. 清除 DMA_CCRx 寄存器中的 DIR 位，根据应用要求可以设置在数据传输完成一半或全部完成时发出中断请求。
6. 设置 DMA_CCRx 寄存器中的 EN 位激活该通道。 当 DMA 控制器中设置的数据传输数目已经完成时，DMA 控制器给 I^2C 接口发送一个传输结束的EOT/ EOT_1 信号。在中断允许的情况下，将产生一个 DMA 中断。

注： 如果使用 DMA 进行接收时，不要设置 I2C_CR2 寄存器的 ITBUFEN 位。

20.3.8 包错误校验(PEC)

包错误校验(PEC)计算器是用于提高通信的可靠性，这个计算器使用下述 CRC-8 多项式对每一位串行数据进行计算：

$$C(x) = x^8 + x^2 + x + 1$$

- PEC 计算由 I2C_CR1 寄存器的 ENPEC 位激活。PEC 使用 CRC-8 算法对所有信息字节进行计算，包括地址和读/写位在内。
  - 在发送时：在最后一个 TxE 事件时设置 I2C_CR1 寄存器的 PEC 传输位，PEC 将在最后一个字节后被发送。
  - 在接收时：在最后一个 RxNE 事件之后设置 I2C_CR1 寄存器的 PEC 位，如果下个接收到的字节不等于内部计算的 PEC，接收器发送一个 NACK。如果是主接收器，不管校对的结果如何，PEC 后都将发送 NACK。PEC 位必须在接收当前字节的 ACK 脉冲之前设置。
- 在 I2C_SR1 寄存器中可获得 PECERR 错误标记/中断。
- 如果 DMA 和 PEC 计算器都被激活：
  - 在发送时：当 I^2C 接口从 DMA 控制器处接收到 EOT 信号时，它在最后一个字节后自动发送 PEC。
  - 在接收时：当 I^2C 接口从 DMA 处接收到一个 EOT_1 信号时，它将自动把下一个字节作为 PEC，并且将检查它。在接收到 PEC 后产生一个 DMA 请求。
- 为了允许中间 PEC 传输，在 I2C_CR2 寄存器中有一个控制位(LAST 位)用于判别是否真是最后一个 DMA 传输。如果确实是最后一个主接收器的 DMA 请求，在接收到最后一个字节后自动发送 NACK。
- 仲裁丢失时 PEC 计算失效。

20.4 I2C 中断请求

下表列出了所有的 I^2C 中断请求

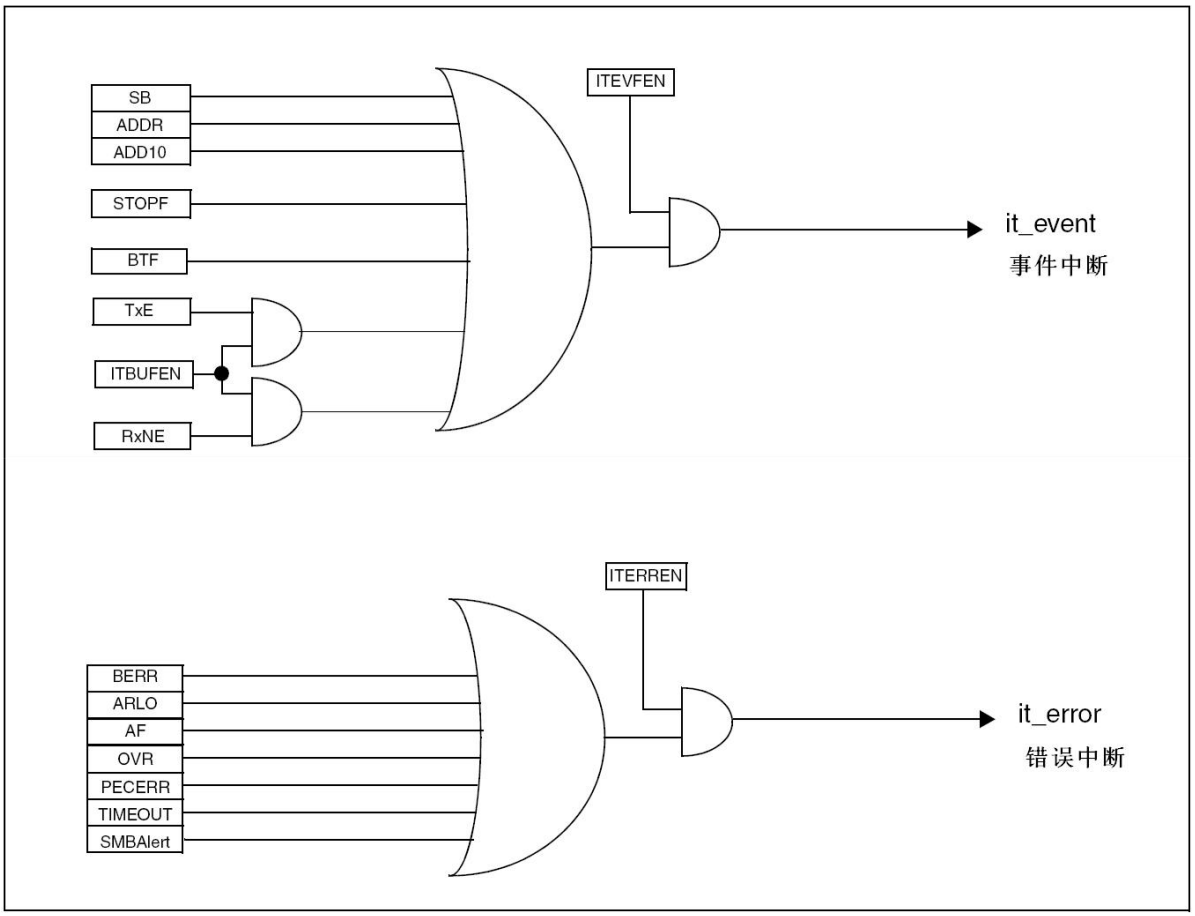
表 79 I2C 中断请求表：

中断事件	事件标志	开启控制位
起始位已发送(主)	SB	ITEVFEN
地址已发送(主) 或地址匹配(从)	ADDR	
10 位头段已发送(主)	ADD10	
已收到停止(从)	STOPF	
数据字节传输完成	BTF	
接收缓冲区非空	RxNE	ITEVFEN 和 ITBUFEN
发送缓冲区空	TxE	
总线错误	BERR	ITERREN
仲裁丢失(主)	ARLO	
响应失败	AF	
过载/欠载	OVR	
PEC 错误	PECERR	
超时/Tlow 错误	TIMEOUT	
SMBus 提醒	SMBALERT	

注： *1. SB、ADDR、ADD10、STOPF、BTF、RxNE 和 TxE 通过逻辑或汇到同一个中断通道中。*

2. BERR、ARLO、AF、OVR、PECERR、TIMEOUT 和 SMBALERT 通过逻辑或汇到同一个中断通道中。

图 172 I2C 中断映射图



20.5 I2C 调试模式

当微控制器进入调试模式(Cortex-M3 核心处于停止状态)时，根据 DBG 模块中的 DBG_I2Cx_SMBUS_TIMEOUT 配置位，SMBUS 超时控制或者继续正常工作或者可以停止。详见第 25.15.2 节。

20.6 I2C 寄存器描述

关于在寄存器描述里面所用到的缩写，详见第 2.1 节。 可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

20.6.1 控制寄存器 1(I2C_CR1)

地址偏移：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	保留	ALERT	PEC	POS	ACK	STOP	START	NO STRETCH	ENGC	ENPEC	ENARP	SMB TYP	保留	SMBUS	PE
rw	res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res	rw	rw

位 15	**SWRST**：软件复位(Software reset) 当被置位时，I^2C 处于复位状态。在复位该位前确信 I^2C 的引脚被释放，总线是空的。 0：I^2C 模块不处于复位状态； 1: I^2C 模块处于复位状态。 注：该位可以用于 BUSY 位为'1'，在总线上又没有检测到停止条件时。
位 14	保留位，硬件强制为 0
位 13	**ALERT**：SMBus 提醒(SMBus alert) 软件可以设置或清除该位；当 PE=0 时，由硬件清除。 0：释放 SMBAlert 引脚使其变高。提醒响应地址头紧跟在 NACK 信号后面； 1：驱动 SMBAlert 引脚使其变低。提醒响应地址头紧跟在 ACK 信号后面。
位 12	**PEC**：数据包出错检测(Packet error checking) 软件可以设置或清除该位；当传送 PEC 后，或起始或停止条件时，或当 PE=0 时硬件将其清除。 0：无 PEC 传输； 1：PEC 传输(在发送或接收模式)。 注：仲裁丢失时，PEC 的计算失效。
位 11	**POS**: 应答/PEC 位置(用于数据接收) (Acknowledge/PEC Position (for data reception)) 软件可以设置或清除该位，或当 PE=0 时，由硬件清除。 0：ACK 位控制当前移位寄存器内正在接收的字节的(N)ACK。 PEC 位表明当前移位寄存器内的字节是 PEC； 1: ACK 位控制在移位寄存器里接收的下一个字节的(N)ACK。PEC 位表明在移位寄存器里接收的下一个字节是 PEC。 注：POS 位只能用在 2 字节的接收配置中，必须在接收数据之前配置。 为了 NACK 第 2 个字节，必须在清除 ADDR 为之后清除 ACK 位。 为了检测第 2 个字节的 PEC，必须在配置了 POS 位之后，拉伸 ADDR 事件时设置 PEC 位。
位 10	**ACK**：应答使能(Acknowledge enable) 软件可以设置或清除该位，或当 PE=0 时，由硬件清除。 0：无应答返回； 1：在接收到一个字节后返回一个应答(匹配的地址或数据)。
位 9	**STOP**：停止条件产生(Stop generation) 软件可以设置或清除该位；或当检测到停止条件时，由硬件清除；当检测到超时错误时，硬件将其置位。 在主模式下： 0：无停止条件产生； 1：在当前字节传输或在当前起始条件发出后产生停止条件。 在从模式下： 0：无停止条件产生；

	1：在当前字节传输或释放 SCL 和 SDA 线。 注：当设置了 STOP、START 或PEC位，在硬件清除这个位之前，软件不要执行任何 I2C_CR1 的写操作；否则有可能会第 2 次设置 STOP、START 或 PEC 位。
位 8	**START**：起始条件产生(Start generation) 软件可以设置或清除该位，或当起始条件发出后或 PE=0 时，由硬件清除。 在主模式下： 0: 无起始条件产生； 1：重复产生起始条件。 在从模式下： 0：无起始条件产生； 1：当总线空闲时，产生起始条件。
位 7	**NOSTRETCH**：禁止时钟延长(从模式) (Clock stretching disable (Slave mode)) 该位用于当 ADDR 或 BTF 标志被置位，在从模式下禁止时钟延长，直到它被软件复位。 0：允许时钟延长； 1：禁止时钟延长。
位 6	**ENGC**：广播呼叫使能(General call enable) 0：禁止广播呼叫。以非应答响应地址 00h； 1：允许广播呼叫. 以应答响应地址 00h。
位 5	**ENPEC**：PEC 使能(PEC enable) 0：禁止 PEC 计算； 1：开启 PEC 计算。
位 4	**ENARP**：ARP 使能(ARP enable) 0：禁止 ARP； 1：使能 ARP。 如果 SMBTYPE=0，使用 SMBus 设备的默认地址。 如果 SMBTYPE=1，使用 SMBus 的主地址。
位 3	**SMBTYPE**：SMBus 类型(SMBus type) 0：SMBus 设备； 1：SMBus 主机。
位 2	保留位，硬件强制为 0。
位 1	**SMBUS**：SMBus 模式(SMBus mode) 0：I2C 模式； 1：SMBus 模式。
位 0	**PE**：I^2C 模块使能(Peripheral enable) 0：禁用 I^2C 模块； 1：启用 I^2C 模块：根据 SMBus 位的设置，相应的 I/O 口需配置为复用功能。 注：如果清除该位时通讯正在进行，在当前通讯结束后，I^2C 模块被禁用并返回空闲状态。 由于在通讯结束后发生 PE＝0，所有的位被清除。 在主模式下，通讯结束之前，绝不能清除该位。

20.6.2 控制寄存器 2(I2C_CR2)

地址偏移：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	保留		FREQ[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

位 15:13	保留位，硬件强制为 0
位 12	**LAST**: DMA 最后一次传输(DMA last transfer) 0：下一次 DMA 的 EOT 不是最后的传输； 1：下一次 DMA 的 EOT 是最后的传输。 注：该位在主接收模式使用，使得在最后一次接收数据时可以产生一个 NACK。
位 11	**DMAEN**：DMA 请求使能(DMA requests enable) 0：禁止 DMA 请求； 1：当 TxE=1 或 RxNE =1 时，允许 DMA 请求。
位 10	**ITBUFEN**：缓冲器中断使能(Buffer interrupt enable) 0：当 TxE=1 或 RxNE=1 时，不产生任何中断； 1：当 TxE=1 或 RxNE=1 时，产生事件中断(不管 DMAEN 是何种状态)。
位 9	**ITEVTEN**：事件中断使能(Event interrupt enable) 0：禁止事件中断； 1：允许事件中断。 在下列条件下，将产生该中断： – SB = 1 (主模式)； – ADDR = 1 (主/从模式)； – ADD10= 1 (主模式)； – STOPF = 1 (从模式)； – BTF = 1，但是没有 TxE 或 RxNE 事件； – 如果 ITBUFEN = 1，TxE 事件为 1； – 如果 ITBUFEN = 1，RxNE 事件为 1。
位 8	**ITERREN**：出错中断使能(Error interrupt enable) 0：禁止出错中断； 1：允许出错中断。 在下列条件下，将产生该中断： – BERR = 1； – ARLO = 1； – AF = 1； – OVR = 1； – PECERR = 1； – TIMEOUT = 1； – SMBAlert = 1。
位 7:6	保留位，硬件强制为 0。
位 5:0	**FREQ[5:0]**：I^2C 模块时钟频率(Peripheral clock frequency) 必须设置正确的输入时钟频率以产生正确的时序，允许的范围在 2～36MHz 之间： 000000：禁用 000001：禁用 000010：2MHz ... 100100：36MHz

	大于 100100：禁用。

20.6.3 自身地址寄存器 1(I2C_OAR1)

复位地址偏移：0x08

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD MODE	保留	保留				ADD[9:8]		ADD[7:1]							ADD0
rw	res	res				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15	**ADDMODE**：寻址模式(从模式) (Addressing mode (slave mode)) 0：7 位从地址(不响应 10 位地址)； 1：10 位从地址(不响应 7 位地址)。
位 14	必须始终由软件保持为'1'。
位 13:10	保留位，硬件强制为 0。
位 9:8	**ADD[9:8]**：接口地址(Interface address) 7 位地址模式时不用关心。 10 位地址模式时为地址的 9~8 位。
位 7:1	**ADD[7:1]**：接口地址(Interface address) 地址的 7~1 位。
位 0	**ADD0**：接口地址(Interface address) 7 位地址模式时不用关心。 10 位地址模式时为地址第 0 位。

20.6.4 自身地址寄存器 2(I2C_OAR2)

地址偏移：0x0C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								ADD2[7:1]							ENDUAL
res								rw	rw	rw	rw	rw	rw	rw	rw

位 15:8	保留位，硬件强制为 0
位 7:1	**ADD2[7:1]**：接口地址(Interface address) 在双地址模式下地址的 7~1 位。
位 0	**ENDUAL**：双地址模式使能位(Dual addressing mode enable) 0：在 7 位地址模式下，只有 OAR1 被识别； 1：在 7 位地址模式下，OAR1 和 OAR2 都被识别。

20.6.5 数据寄存器(I2C_DR)

地址偏移：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								DR[7:0]							
res								rw	rw	rw	rw	rw	rw	rw	rw

位 15:8	保留位，硬件强制为 0
位 7:0	**DR[7:0]**：8 位数据寄存器(8-bit data register) 用于存放接收到的数据或放置用于发送到总线的数据 发送器模式：当写一个字节至 DR 寄存器时，自动启动数据传输。一旦传输开始(TxE=1)，如果能及时把下一个需传输的数据写入 DR 寄存器，I^2C 模块将保持连续的数据流。 接收器模式：接收到的字节被拷贝到 DR 寄存器(RxNE=1)。在接收到下一个字节(RxNE=1)之前读出数据寄存器，即可实现连续的数据传送。 注：在从模式下，地址不会被拷贝进数据寄存器 DR； 注：硬件不管理写冲突(如果 TxE=0，仍能写入数据寄存器)； 注：如果在处理 ACK 脉冲时发生 ARLO 事件，接收到的字节不会被拷贝到数据寄存器里，因此不能读到它。

20.6.6 状态寄存器 1(I2C_SR1)

地址偏移：0x14

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMB ALERT	TIM E	保留	PE C	OVR	AF	ARLO	BERR	TxE	RxNE	保留	STOPF	ADD10	BTF	ADDR	SB
rc w0	rc w0	res	rc w0	rc w0	rc w0	rc w0	rc w0	r	r	res	r	r	r	r	r

位 15	**SMBALERT**：SMBus 提醒(SMBus alert) 在 SMBus 主机模式下： 0：无 SMBus 提醒； 1：在引脚上产生 SMBAlert 提醒事件。 在 SMBus 从机模式下： 0：没有 SMBAlert 响应地址头序列； 1：收到 SMBAlert 响应地址头序列至 SMBAlert 变低。 – 该位由软件写'0'清除，或在 PE=0 时由硬件清除。
位 14	**TIMEOUT**：超时或 Tlow 错误(Timeout or Tlow error) 0：无超时错误； 1：SCL 处于低已达到25ms(超时)；或者主机低电平累积时钟扩展时间超 10ms(Tlow:mext)；或从设备低电平累积时钟扩展时间超过 25ms(Tlow:sext)。 – 当在从模式下设置该位：从设备复位通讯，硬件释放总线。 – 当在主模式下设置该位：硬件发出停止条件。 – 该位由软件写'0'清除，或在 PE=0 时由硬件清除。
位 13	保留位，硬件强制为 0。
位 12	**PECERR**：在接收时发生 PEC 错误(PEC Error in reception) 0: 无 PEC 错误：接收到 PEC 后接收器返回 ACK(如果 ACK=1)； 1: 有 PEC 错误：接收到 PEC 后接收器返回 NACK(不管 ACK 是什么值)。 – 该位由软件写'0'清除，或在 PE=0 时由硬件清除。
位 11	**OVR**：过载/欠载(Overrun/Underrun) 0: 无过载/欠载； 1: 出现过载/欠载。 – 当 NOSTRETCH=1 时，在从模式下该位被硬件置位，同时： – 在接收模式中当收到一个新的字节时(包括 ACK 应答脉冲)，数据寄存器里的内容还未被出，则新接收的字节将丢失。 – 在发送模式中当要发送一个新的字节时，却没有新的数据写入数据寄存器，同样的字节将被发送两次。 – 该位由软件写'0'清除，或在 PE=0 时由硬件清除。。 注：如果数据寄存器的写操作发生时间非常接近 SCL 的上升沿，发送的数据是不确定的，并发生保持时间错误。
位 10	**AF**：应答失败(Acknowledge failure) 0：没有应答失败； 1：应答失败。 – 当没有返回应答时，硬件将置该位为'1'。 – 该位由软件写'0'清除，或在 PE=0 时由硬件清除。

位 9	**ARLO**：仲裁丢失(主模式) (Arbitration lost (master mode)) 0：没有检测到仲裁丢失； 1：检测到仲裁丢失。 当接口失去对总线的控制给另一个主机时，硬件将将置该位为'1'。 – 该位由软件写'0'清除，或在 PE=0 时由硬件清除。 在 ARLO 事件之后，I^2C 接口自动切换回从模式(M/SL=0)。 注：在 SMBUS 模式下，在从模式下对数据的仲裁仅仅发生在数据阶段，或应答传输区间(不包括地址的应答)。
位 8	**BERR**：总线出错(Bus error) 0：无起始或停止条件出错； 1：起始或停止条件出错。 – 当接口检测到错误的起始或停止条件，硬件将该位置'1'。 – 该位由软件写'0'清除，或在 PE=0 时由硬件清除。
位 7	**TxE**：数据寄存器为空(发送时) (Data register empty (transmitters)) 0：数据寄存器非空； 1：数据寄存器空。 – 在发送数据时，数据寄存器为空时该位被置'1'，在发送地址阶段不设置该位。 – 软件写数据到 DR 寄存器可清除该位；或在发生一个起始或停止条件后，或当 PE=0 时由硬自动清除。 如果收到一个 NACK，或下一个要发送的字节是 PEC(PEC=1)，该位不被置位。 注：在写入第 1 个要发送的数据后，或设置了 BTF 时写入数据，都不能清除 TxE 位，这是因数据寄存器仍然为空。
位 6	**RxNE**：数据寄存器非空(接收时) (Data register not empty (receivers)) 0：数据寄存器为空； 1：数据寄存器非空。 – 在接收时，当数据寄存器不为空，该位被置'1'。在接收地址阶段，该位不被置位。 – 软件对数据寄存器的读写操作清除该位，或当 PE=0 时由硬件清除。 在发生 ARLO 事件时，RxNE 不被置位。 注：当设置了 BTF 时，读取数据不能清除 RxNE 位，因为数据寄存器仍然为满。
位 5	保留位，硬件强制为 0
位 4	**STOPF**：停止条件检测位(从模式) (Stop detection (slave mode)) 0：没有检测到停止条件； 1：检测到停止条件。 – 在一个应答之后(如果 ACK=1)，当从设备在总线上检测到停止条件时，硬件将该位置'1'。 – 软件读取 SR1 寄存器后，对 CR1 寄存器的写操作将清除该位，或当 PE=0 时，硬件清位。 注：在收到 NACK 后，STOPF 位不被置位。
位 3	**ADD10**：10 位头序列已发送(主模式) (10-bit header sent (Master mode)) 0：没有 ADD10 事件发生； 1：主设备已经将第一个地址字节发送出去。 – 在 10 位地址模式下，当主设备已经将第一个字节发送出去时，硬件将该位置'1'。 – 软件读取 SR1 寄存器后，对 CR1 寄存器的写操作将清除该位，或当 PE=0 时，硬件清位。 注：收到一个 NACK 后，ADD10 位不被置位。

位 2	**BTF**：字节发送结束(Byte transfer finished) 0：字节发送未完成； 1：字节发送结束。 当 NOSTRETCH=0 时，在下列情况下硬件将该位置’1’： – 在接收时，当收到一个新字节(包括 ACK 脉冲)且数据寄存器还未被读取(RxNE=1)。 – 在发送时，当一个新数据将被发送且数据寄存器还未被写入新的数据(TxE=1)。 – 在软件读取 SR1 寄存器后，对数据寄存器的读或写操作将清除该位；或在传输中发送一个起始或停止条件后，或当 PE=0 时，由硬件清除该位。 注：在收到一个 NACK 后，BTF 位不会被置位。 如果下一个要传输的字节是 PEC(I2C_SR2 寄存器中 TRA 为’1’，同时 I2C_CR1 寄存器中为’1’)，BTF 位不会被置位。
位 1	**ADDR**：地址已被发送(主模式)/地址匹配(从模式) (Address sent (master (slave mode)) 在软件读取 SR1 寄存器后，对 SR2 寄存器的读操作将清除该位，或当 PE=0 时，由硬件清除位。 **地址匹配(从模式)** 0：地址不匹配或没有收到地址； 1：收到的地址匹配。 – 当收到的从地址与 OAR 寄存器中的内容相匹配、或发生广播呼叫、或 SMBus 设备默认地或 SMBus 主机识别出 SMBus 提醒时，硬件就将该位置’1’(当对应的设置被使能时)。 **地址已被发送(主模式)** 0：地址发送没有结束； 1：地址发送结束。 – 10 位地址模式时，当收到地址的第二个字节的 ACK 后该位被置’1’。 – 7 位地址模式时，当收到地址的 ACK 后该位被置’1’。 注：在收到 NACK 后，ADDR 位不会被置位。
位 0	**SB**：起始位(主模式) (Start bit (Master mode)) 0：未发送起始条件； 1：起始条件已发送。 – 当发送出起始条件时该位被置’1’。 – 软件读取 SR1 寄存器后，写数据寄存器的操作将清除该位，或当 PE=0 时，硬件清除该位。

20.6.7 状态寄存器 2 (I2C_SR2)

地址偏移：0x18

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEC[7:0]								DUALF	SMB HOS	SMB DEFAU LT	GEN CAL	保留	TRA	BUSY	MSL
r	r	r	r	r	r	r	r	r	r	r	r	res	r	r	r

位 15:8	**PEC[7:0]**：数据包出错检测(Packet error checking register) 当 ENPEC=1 时，PEC[7:0]存放内部的 PEC 的值。
位 7	**DUALF**：双标志(从模式) (Dual flag (Slave mode)) 0：接收到的地址与 OAR1 内的内容相匹配； 1：接收到的地址与 OAR2 内的内容相匹配。 – 在产生一个停止条件或一个重复的起始条件时，或 PE=0 时，硬件将该位清除。
位 6	**SMBHOST**: SMBus 主机头系列(从模式) (SMBus host header (Slave mode)) 0：未收到 SMBus 主机的地址； 1：当 SMBTYPE=1 且 ENARP=1 时，收到 SMBus 主机地址。 – 在产生一个停止条件或一个重复的起始条件时，或 PE=0 时，硬件将该位清除。
位 5	**SMBDEFAULT**：SMBus 设备默认地址(从模式) (SMBus device default address (Slave mode)) 0：未收到 SMBus 设备的默认地址； 1：当 ENARP=1 时，收到 SMBus 设备的默认地址。 – 在产生一个停止条件或一个重复的起始条件时，或 PE=0 时，硬件将该位清除。
位 4	**GENCALL**：广播呼叫地址(从模式) (General call address (Slave mode)) 0：未收到广播呼叫地址； 1：当 ENGC=1 时，收到广播呼叫的地址。 – 在产生一个停止条件或一个重复的起始条件时，或 PE=0 时，硬件将该位清除。
位 3	保留位，硬件强制为 0
位 2	**TRA**：发送/接收(Transmitter/receiver) 0：接收到数据； 1：数据已发送； 在整个地址传输阶段的结尾，该位根据地址字节的 R/W 位来设定。 在检测到停止条件(STOPF=1)、重复的起始条件或总线仲裁丢失(ARLO=1)后，或当 PE=0 时，硬件将其清除。
位 1	**BUSY**：总线忙(Bus busy) 0：在总线上无数据通讯； 1：在总线上正在进行数据通讯。 – 在检测到 SDA 或 SCl 为低电平时，硬件将该位置'1'； – 当检测到一个停止条件时，硬件将该位清除。 该位指示当前正在进行的总线通讯，当接口被禁用(PE=0)时该信息仍然被更新。
位 0	**MSL**：主从模式(Master/slave) 0：从模式； 1：主模式。 – 当接口处于主模式(SB=1)时，硬件将该位置位； – 当总线上检测到一个停止条件、仲裁丢失(ARLO=1 时)、或当 PE=0 时，硬件清除该位。

20.6.8 时钟控制寄存器(I2C_CCR)

地址偏移：0x1C

复位值：0x0000

注： *1. 要求 FPCLK1 应当是 10 MHz 的整数倍，这样可以正确地产生 400KHz 的快速时钟。*
2. CCR 寄存器只有在关闭 I2C 时(PE=0)才能设置

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S	DUTY	保留		CCR[11:0]											
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15	**F/S**：I^2C 主模式选项(I^2C master mode selection) 0：标准模式的 I^2C； 1：快速模式的 I^2C。
位 14	**DUTY**：快速模式时的占空比(Fast mode duty cycle) 0：快速模式下：T_{low}/T_{high} = 2； 1: 快速模式下：T_{low}/T_{high} = 16/9(见 CCR)。
位 13:12	保留位，硬件强制为 0。
位 11:0	**CCR[11:0]**：快速/标准模式下的时钟控制分频系数(主模式) (Clock control register in Fast/Standard mode (Master mode)) 该分频系数用于设置主模式下的 SCL 时钟。 在 I^2C 标准模式或 SMBus 模式下： T_{high} = CCR×T_{PCLK1} T_{low} = CCR ×T_{PCLK1} 在 I^2C 快速模式下： 如果 DUTY = 0： T_{high} = CCR ×T_{PCLK1} T_{low} = 2 × CCR × T_{PCLK1} 如果 DUTY = 1：(速度达到 400kHz) T_{high} = 9 × CCR ×T_{PCLK1} T_{low} = 16 × CCR ×T_{PCLK1} 例如：在标准模式下，产生 100kHz 的 SCL 的频率： 如果 FREQR = 08，T_{PCLK1} = 125ns，则 CCR 必须写入 0x28(40×125ns = 5000 ns)。 注： 1. 允许设定的最小值为 0x04，在快速 DUTY 模式下允许的最小值为 0x01； 2. $T_{high}=t_{r(SCL)}+t_{w(SCLH)}$，详见数据手册中对这些参数的定义； 3. $T_{low}=t_{f(SCL)}+t_{w(SCLL)}$，详见数据手册中对这些参数的定义； 4. 这些延时没有过滤器； 5. 只有在关闭 I^2C 时(PE = 0)才能设置 CCR 寄存器； 6. f_{CK} 应当是 10MHz 的整数倍，这样可以正确产生 400kHz 的快速时钟。

20.6.9 TRISE 寄存器(I2C_TRISE)

地址偏移：0x20

复位值：0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										TRISE[5:0]					
res										rw	rw	rw	rw	rw	rw

位 15:6	保留位，硬件强制为 0
位 5:0	**TRISE[5:0]** ：在快速/标准模式下的最大上升时间(主模式) (Maximum rise time in Fast/Standard mode (Master mode)) 这些位必须设置为 I^2C 总线规范里给出的最大的 SCL 上升时间，增长步幅为 1。 例如：标准模式中最大允许 SCL 上升时间为 1000ns。如果在 I2C_CR2 寄存器中 FREQ[5:0] 中的 值等于 0x08 且 T_{PCLK1}=125ns，故 TRISE[5:0]中必须写入 09h(1000ns/125 ns =8+1)。滤波器的值也可以加到 TRISE[5:0]内。 如果结果不是一个整数，则将整数部分写入 TRISE[5:0]以确保 t_{HIGH} 参数。 注：只有当 I^2C 被禁用(PE=0)时，才能设置 TRISE[5:0]。

20.6.10 I2C 寄存器地址映象

表 80 I2C 寄存器地址映象和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	I2C_CR1	保留																SWRST	保留	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	保留	SMBUS	PE
	复位值																	0		0	0	0	0	0	0	0	0	0	0	0		0	0
0x04	I2C_CR2	保留														ADDMODE					LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	保留	FREQ[5:0]						
	复位值															0					0	0	0	0	0		0	0	0	0	0	0	0
0x08	I2C_OAR1	保留														ADD[9:8]		保留		保留						ADD[7:1]							ADD0
	复位值															0		1						0	0	0	0	0	0	0	0		ENDUAL
0x0C	I2C_OAR2	保留																								ADD2[7:1]							
	复位值																									0	0	0	0	0	0	0	0
0x10	I2C_DR	保留																								DR[7:0]							
	复位值																									0	0	0	0	0	0	0	0
0x14	I2C_SR1	保留																SMBALERT	TIMEOUT	保留	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	保留	STOPF	ADD10	BTF	ADDR	SB
	复位值																	0	0		0	0	0	0	0	0	0		0	0	0	0	0
0x18	I2C_SR2	保留																PEC[7:0]								DUALF	SMBHOST	SMBDEFAU	GENCALL	保留	TRA	BUSY	MSL
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x1C	I2C_CCR	保留																F/S	DUTY	保留	CCR[11:0]												
	复位值																	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	I2C_TRISE	保留																												TRISE[5:0]			
	复位值																													0	0	0	0	1	0

关于寄存器起始地址，参见表 1。

21 通用同步异步收发器（USART）

21.1 USART 介绍

通用同步异步收发器(USART)提供了一种灵活的方法与使用工业标准 NRZ 异步串行数据格式的外部设备之间进行全双工数据交换。USART 利用分数波特率发生器提供宽范围的波特率选择。

它支持同步单向通信和半双工单线通信，也支持 LIN(局部互连网)，智能卡协议和 IrDA(红外数据组织)SIR ENDEC 规范，以及调制解调器(CTS/RTS)操作。它还允许多处理器通信。

使用多缓冲器配置的 DMA 方式，可以实现高速数据通信。

21.2 USART 主要特性

- 全双工的，异步通信
- NRZ 标准格式
- 分数波特率发生器系统
  - 发送和接收共用的可编程波特率，最高达 4.5Mbits/s
- 可编程数据字长度(8 位或 9 位)
- 可配置的停止位-支持 1 或 2 个停止位
- LIN 主发送同步断开符的能力以及 LIN 从检测断开符的能力
  - 当 USART 硬件配置成 LIN 时，生成 13 位断开符；检测 10/11 位断开符
- 发送方为同步传输提供时钟
- IRDA SIR 编码器解码器
  - 在正常模式下支持 3/16 位的持续时间
- 智能卡模拟功能
  - 智能卡接口支持 ISO7816-3 标准里定义的异步智能卡协议
  - 智能卡用到的 0.5 和 1.5 个停止位
- 单线半双工通信
- 可配置的使用 DMA 的多缓冲器通信
  - 在 SRAM 里利用集中式 DMA 缓冲接收/发送字节
- 单独的发送器和接收器使能位
- 检测标志
  - 接收缓冲器满
  - 发送缓冲器空
  - 传输结束标志
- 校验控制
  - 发送校验位
  - 对接收数据进行校验
- 四个错误检测标志
  - 溢出错误
  - 噪音错误
  - 帧错误
  - 校验错误
- 10 个带标志的中断源
  - CTS 改变
  - LIN 断开符检测

- 发送数据寄存器空
- 发送完成
- 接收数据寄存器满
- 检测到总线为空闲
- 溢出错误
- 帧错误
- 噪音错误
- 校验错误

- 多处理器通信 -- 如果地址不匹配，则进入静默模式
- 从静默模式中唤醒(通过空闲总线检测或地址标志检测)
- 两种唤醒接收器的方式：地址位(MSB，第 9 位)，总线空闲

21.3 USART 功能概述

接口通过三个引脚与其他设备连接在一起(见图 173)。任何 USART 双向通信至少需要两个脚：接收数据输入(RX)和发送数据输出(TX)。

RX：接收数据串行输。通过过采样技术来区别数据和噪音，从而恢复数据。

TX：发送数据输出。当发送器被禁止时，输出引脚恢复到它的 I/O 端口配置。当发送器被激活，并且不发送数据时，TX 引脚处于高电平。在单线和智能卡模式里，此 I/O 口被同时用于数据的发送和接收（在 USART 层次，数据在 SW_RX 上收到）。

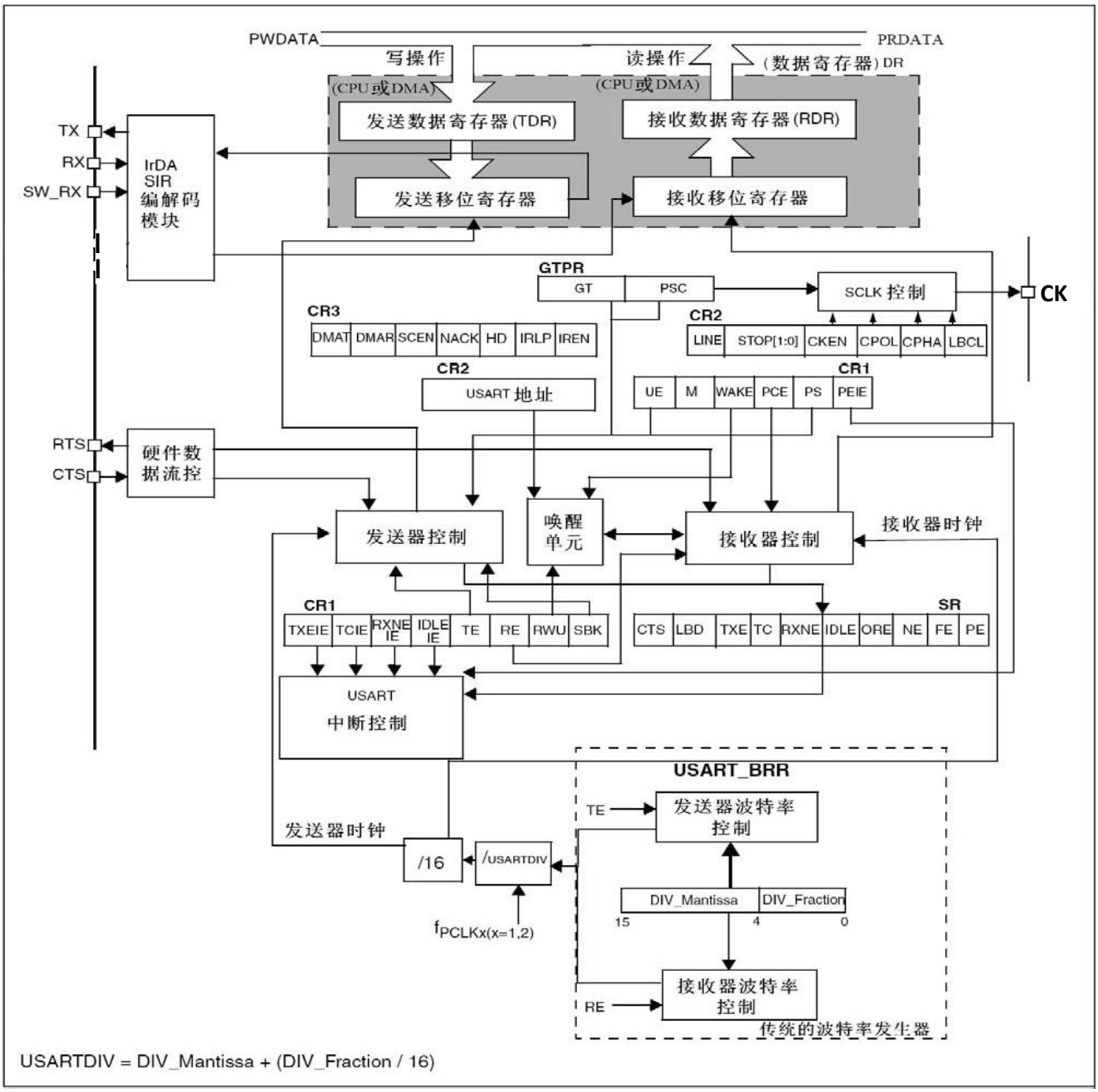
- 总线在发送或接收前应处于空闲状态
- 一个起始位
- 一个数据字(8 或 9 位)，最低有效位在前
- 0.5，1.5，2 个的停止位，由此表明数据帧的结束
- 使用分数波特率发生器 —— 12 位整数和 4 位小数的表示方法。
- 一个状态寄存器(USART_SR)
- 数据寄存器(USART_DR)
- 一个波特率寄存器(USART_BRR)，12 位的整数和 4 位小数
- 一个智能卡模式下的保护时间寄存器(USART_GTPR) 关于以上寄存器中每个位的具体定义，请参考寄存器描述第 22.6 节：USART 寄存器描述。 在同步模式中需要下列引脚：

- **CK**：发送器时钟输出。此引脚输出用于同步传输的时钟， (在 Start 位和 Stop 位上没有时钟脉冲，软件可选地，可以在最后一个数据位送出一个时钟脉冲)。数据可以在 RX 上同步被接收。这可以用来控制带有移位寄存器的外部设备(例如 LCD 驱动器)。时钟相位和极性都是软件可编程的。在智能卡模式里，CK 可以为智能卡提供时钟。

下列引脚在硬件流控模式中需要：

- **CTS:** 清除发送，若是高电平，在当前数据传输结束时阻断下一次的数据发送。
- **RTS:** 发送请求，若是低电平，表明 USART 准备好接收数据

图 173 USART 框图



21.3.1 USART 特性描述

字长可以通过编程 USART_CR1 寄存器中的 M 位，选择成 8 或 9 位(见图 174)。在起始位期间，TX 脚处于低电平，在停止位期间处于高电平。

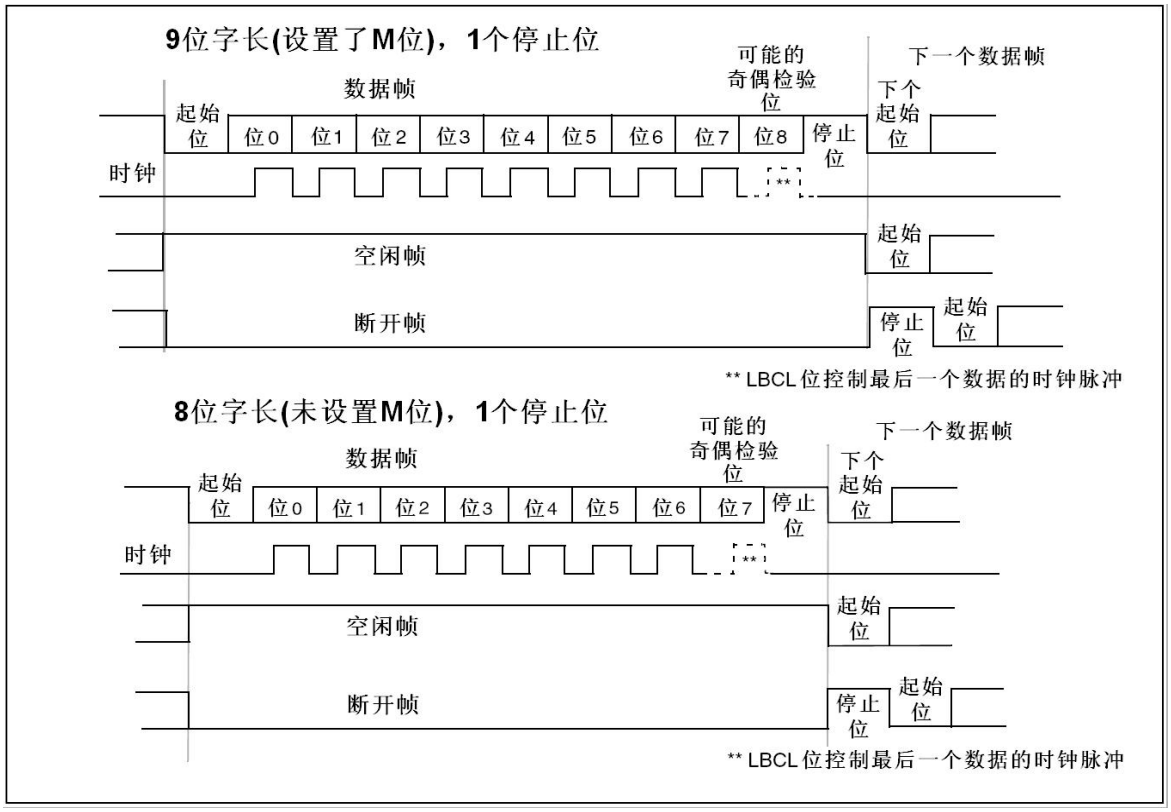
空闲符号被视为完全由'1'组成的一个完整的数据帧，后面跟着包含了数据的下一帧的开始位('1' 的位数也包括了停止位的位数)。

断开符号 被视为在一个帧周期内全部收到'0'(包括停止位期间，也是'0')。在断开帧结束时，发送器再插入 1 或 2 个停止位('1')来应答起始位。

发送和接收由一共用的波特率发生器驱动，当发送器和接收器的使能位分别置位时，分别为其产生时钟。

随后将有每个功能块的详细说明。

图 174 字长设置



21.3.2 发送器

发送器根据 M 位的状态发送 8 位或 9 位的数据字。当发送使能位(TE)被设置时，发送移位寄存器 中的数据在 TX 脚上输出，相应的时钟脉冲在 CK 脚上输出。

字符发送

在 USART 发送期间，在 TX 引脚上首先移出数据的最低有效位。在此模式里，USART_DR 寄存器包含了一个内部总线和发送移位寄存器之间的缓冲器(见图 173)。

每个字符之前都有一个低电平的起始位；之后跟着的停止位，其数目可配置。

USART 支持多种停止位的配置：0.5、1、1.5 和 2 个停止位。

注： 1．在数据传输期间不能复位 TE 位，否则将破坏 TX 脚上的数据，因为波特率计数器停止计数。正在传输的当前数据将丢失。

2． TE 位被激活后将发送一个空闲帧。

可配置的停止位

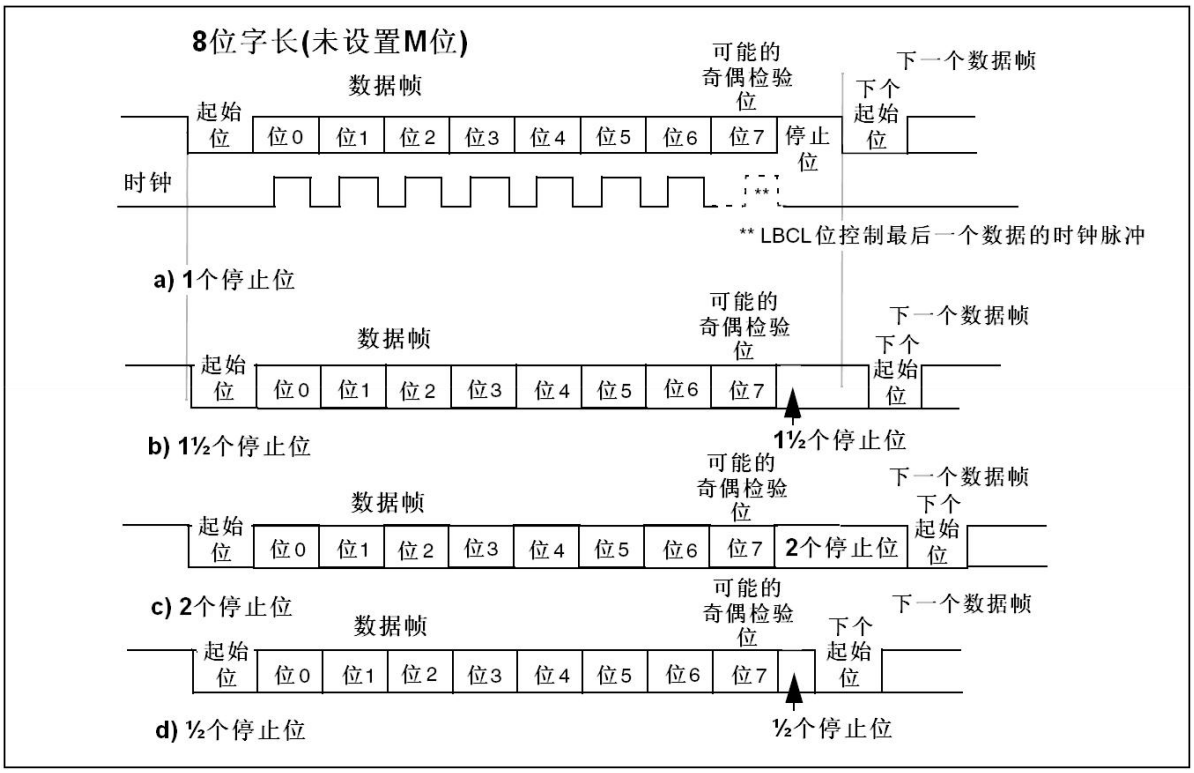
随每个字符发送的停止位的位数可以通过控制寄存器 2 的位 13、12 进行编程。

1. 1 个停止位：停止位位数的默认值。
2. 2 个停止位：可用于常规 USART 模式、单线模式以及调制解调器模式。
3. 0.5 个停止位：在智能卡模式下接收数据时使用。
4. 1.5 个停止位：在智能卡模式下发送和接收数据时使用。

空闲帧包括了停止位。

断开帧是 10 位低电平，后跟停止位(当 m=0 时)；或者 11 位低电平，后跟停止位(m=1 时)。不可能传输更长的断开帧(长度大于 10 或者 11 位)。

图 175 配置停止位



配置步骤：

1. 通过在 USART_CR1 寄存器上置位 UE 位来激活 USART
2. 编程 USART_CR1 的 M 位来定义字长。
3. 在 USART_CR2 中编程停止位的位数。
4. 如果采用多缓冲器通信，配置 USART_CR3 中的 DMA 使能位(DMAT)。按多缓冲器通信中的描述配置 DMA 寄存器。
5. 利用 USART_BRR 寄存器选择要求的波特率。
6. 设置 USART_CR1 中的 TE 位，发送一个空闲帧作为第一次数据发送。
7. 把要发送的数据写进 USART_DR 寄存器(此动作清除 TXE 位)。在只有一个缓冲器的情况下，对每个待发送的数据重复步骤 7。
8. 在 USART_DR 寄存器中写入最后一个数据字后，要等待 TC=1，它表示最后一个数据帧的传输结束。当需要关闭 USART 或需要进入停机模式之前，需要确认传输结束，避免破坏最后一次传输。

单字节通信

清零 TXE 位总是通过对数据寄存器的写操作来完成的。TXE 位由硬件来设置，它表明：

- 数据已经从 TDR 移送到移位寄存器，数据发送已经开始
- TDR 寄存器被清空
- 下一个数据可以被写进 USART_DR 寄存器而不会覆盖先前的数据

如果 TXEIE 位被设置，此标志将产生一个中断。

如果此时 USART 正在发送数据，对 USART_DR 寄存器的写操作把数据存进 TDR 寄存器，并在当 前传输结束时把该数据复制进移位寄存器。

如果此时 USART 没有在发送数据，处于空闲状态，对 USART_DR 寄存器的写操作直接把数据放 进移位寄存器，数据传输开始，TXE 位立即被置起。

当一帧发送完成时(停止位发送后)并且设置了 TXE 位，TC 位被置起，如果 USART_CR1 寄存器中的 TCIE 位被置起时，则会产生中断。

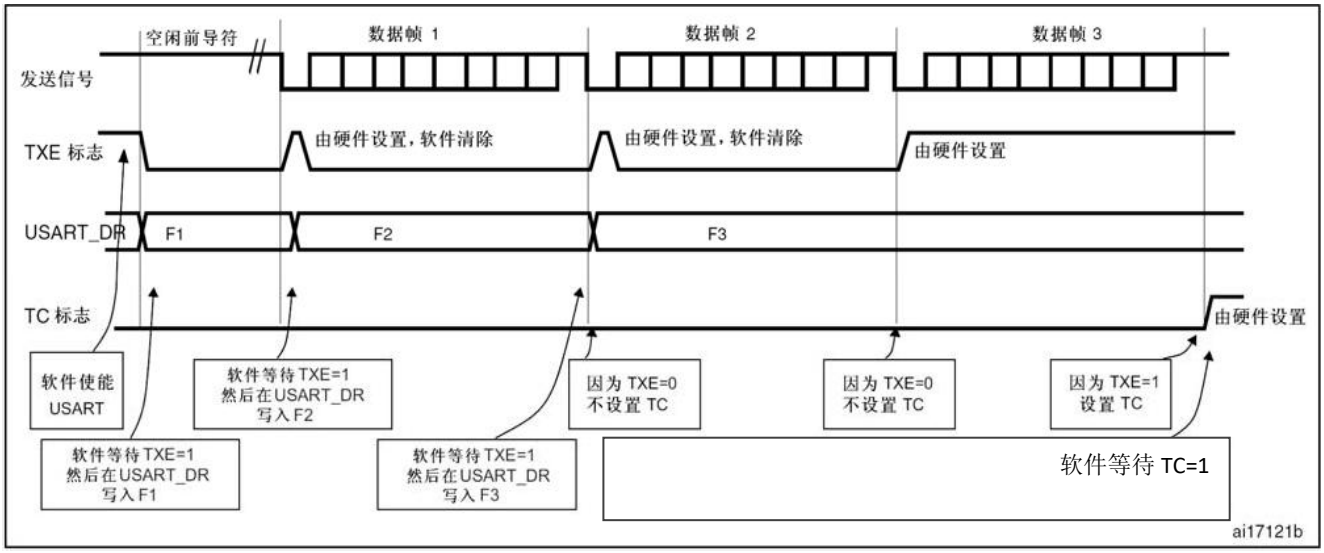
在 USART_DR 寄存器中写入了最后一个数据字后，在关闭 USART 模块之前或设置微控制器进入低功耗模式(详见下图)之前，必须先等待 TC=1。

使用下列软件过程清除 TC 位：

1. 读一次 USART_SR 寄存器；
2. 写一次 USART_DR 寄存器。

注：　TC 位也可以通过软件对它写'0'来清除。此清零方式只推荐在多缓冲器通信模式下使用。

图 176 发送时 TC/TXE 的变化情况



断开符号

设置 SBK 可发送一个断开符号。断开帧长度取决 M 位(见图 174)。如果设置 SBK=1，在完成当前数据发送后，将在 TX 线上发送一个断开符号。断开字符发送完成时(在断开符号的停止位时)SBK 被硬件复位。USART 在最后一个断开帧的结束处插入一逻辑'1'，以保证能识别下一帧的起始位。

注意：　如果在开始发送断开帧之前，软件又复位了 SBK 位，断开符号将不被发送。如果要发送两个连续的断开帧，SBK 位应该在前一个断开符号的停止位之后置起。

空闲符号

置位 TE 将使得 USART 在第一个数据帧前发送一空闲帧。

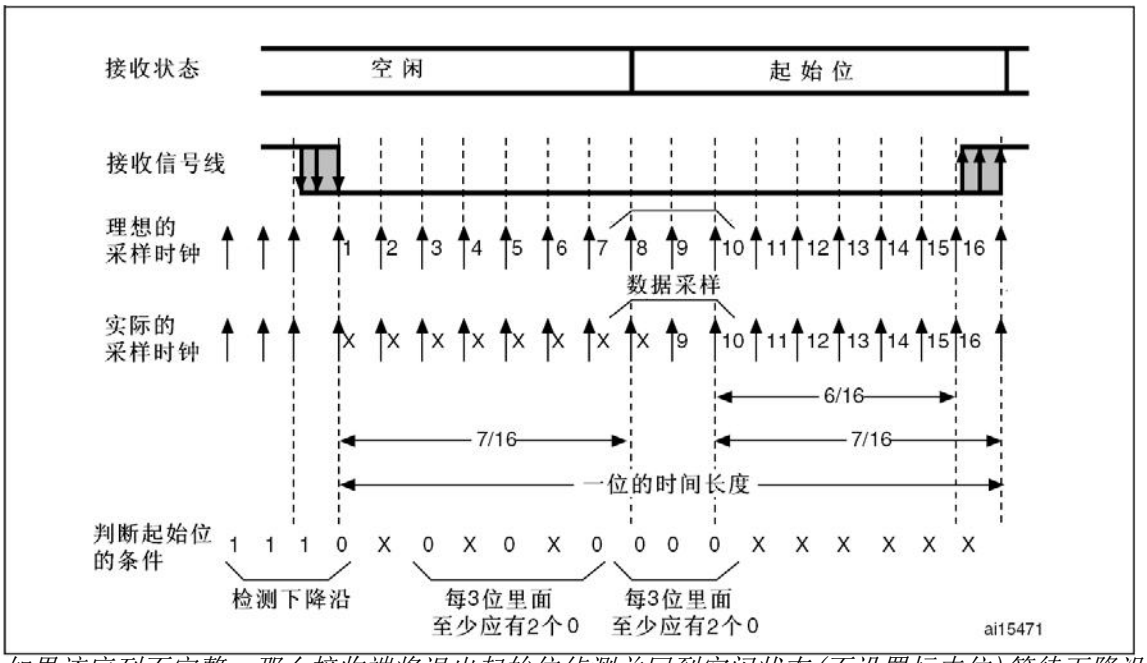
21.3.3 接收器

USART 可以根据 USART_CR1 的 M 位接收 8 位或 9 位的数据字。

起始位侦测

在 USART 中，如果辨认出一个特殊的采样序列，那么就认为侦测到一个起始位。 该序列为：1 1 1 0 X 0 X 0 X 0 0 0 0

图 177 起始位侦测



注意： *如果该序列不完整，那么接收端将退出起始位侦测并回到空闲状态(不设置标志位)等待下降沿。如果3个采样点都为’0’(在第3、5、7位的第一次采样，和在第8、9、10的第二次采样都为’0’)，则确认收到起始位，这时设置 RXNE 标志位，如果 RXNEIE=1，则产生中断。*

如果两次 3 个采样点上仅有 2 个是’0’(第 3、5、7 位的采样点和第 8、9、10 位的采样点)，那么起始位仍然是有效的，但是会设置 NE 噪声标志位。如果不能满足这个条件，则中止起始位的侦测过程，接收器会回到空闲状态(不设置标志位)。

如果有一次 3 个采样点上仅有 2 个是’0’(第 3、5、7 位的采样点或第 8、9、10 位的采样点)，那么起始位仍然是有效的，但是会设置 NE 噪声标志位。

字符接收

在 USART 接收期间，数据的最低有效位首先从 RX 脚移进。在此模式里，USART_DR 寄存器包含的缓冲器位于内部总线和接收移位寄存器之间。

配置步骤：

1. 将 USART_CR1 寄存器的 UE 置 1 来激活 USART。
2. 编程 USART_CR1 的 M 位定义字长
3. 在 USART_CR2 中编写停止位的个数
4. 如果需多缓冲器通信，选择 USART_CR3 中的 DMA 使能位(DMAR)。按多缓冲器通信所要求的配置 DMA 寄存器。
5. 利用波特率寄存器 USART_BRR 选择希望的波特率。
6. 设置 USART_CR1 的 RE 位。激活接收器，使它开始寻找起始位。

当一字符被接收到时，

- RXNE 位被置位。它表明移位寄存器的内容被转移到 RDR。换句话说，数据已经被接收并且可以被读出(包括与之有关的错误标志)。
- 如果 RXNEIE 位被设置，产生中断。
- 在接收期间如果检测到帧错误，噪音或溢出错误，错误标志将被置起，
- 在多缓冲器通信时，RXNE 在每个字节接收后被置起，并由 DMA 对数据寄存器的读操作而清零。
- 在单缓冲器模式里，由软件读 USART_DR 寄存器完成对 RXNE 位清除。RXNE 标志也可以通过对它写 0 来清除。RXNE 位必须在下一字符接收结束前被清零，以避免溢出错误。

注意： 在接收数据时，RE 位不应该被复位。如果 RE 位在接收时被清零，当前字节的接收被丢失。

断开符号

当接收到一个断开帧时，USART 像处理帧错误一样处理它。

空闲符号

当一空闲帧被检测到时，其处理步骤和接收到普通数据帧一样，但如果 IDLEIE 位被设置将产生一个中断。

溢出错误

如果 RXNE 还没有被复位，又接收到一个字符，则发生溢出错误。数据只有当 RXNE 位被清零后才能从移位寄存器转移到 RDR 寄存器。RXNE 标记是接收到每个字节后被置位的。如果下一个数据已被收到或先前 DMA 请求还没被服务时，RXNE 标志仍是置起的，溢出错误产生。

当溢出错误产生时：

- ORE 位被置位。
- RDR 内容将不会丢失。读 USART_DR 寄存器仍能得到先前的数据。
- 移位寄存器中以前的内容将被覆盖。随后接收到的数据都将丢失。
- 如果 RXNEIE 位被设置或 EIE 和 DMAR 位都被设置，中断产生。
- 顺序执行对 USART_SR 和 USART_DR 寄存器的读操作，可复位 ORE 位

注意： 当 ORE 位置位时，表明至少有 1 个数据已经丢失。

有两种可能性：

- 如果 RXNE=1，上一个有效数据还在接收寄存器 RDR 上，可以被读出。
- 如果 RXNE=0，这意味着上一个有效数据已经被读走，RDR 已经没有东西可读。当上一个有效数据在 RDR 中被读取的同时又接收到新的(也就是丢失的)数据时，此种情况可能发生。在读序列期间(在 USART_SR 寄存器读访问和 USART_DR 读访问之间)接收到新的数据，此种情况也可能发生。

噪音错误

使用过采样技术(同步模式除外)，通过区别有效输入数据和噪音来进行数据恢复。

图 178 检测噪声的数据采样

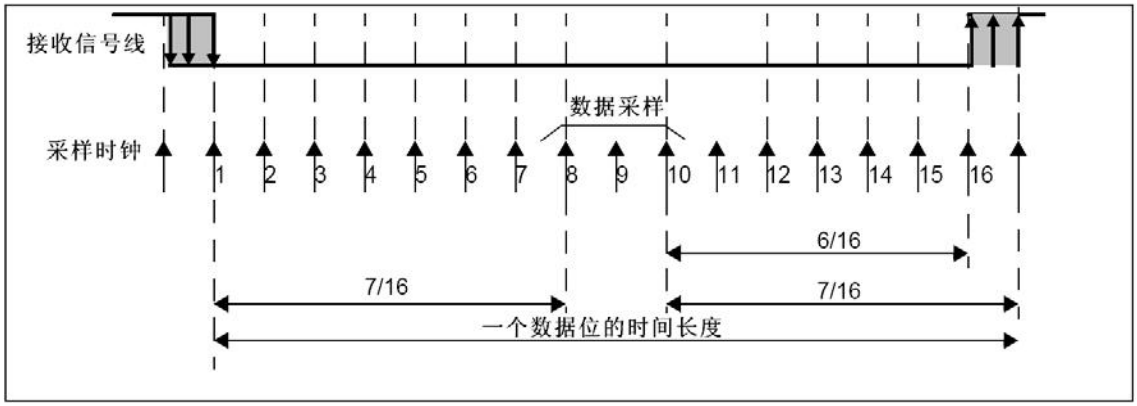


表 81 检测噪声的数据采样

采样值	NE 状态	接收的位	数据有效性
000	0	0	有效
001	1	0	无效
010	1	0	无效
011	1	1	无效
100	1	0	无效
101	1	1	无效
110	1	1	无效
111	0	1	有效

当在接收帧中检测到噪音时：

- 在 RXNE 位的上升沿设置 NE 标志。
- 无效数据从移位寄存器传送到 USART_DR 寄存器。
- 在单个字节通信情况下，没有中断产生。然而，因为 NE 标志位和 RXNE 标志位是同时被设置，RXNE 将产生中断。在多缓冲器通信情况下，如果已经设置了 USART_CR3 寄存器中 EIE 位，将产生一个中断。

先读出 USART_SR，再读出 USART_DR 寄存器，将清除 NE 标志位

帧错误

当以下情况发生时检测到帧错误：由于没有同步上或大量噪音的原因，停止位没有在预期的时间上接和收识别出来。当帧错误被检测到时：

- FE 位被硬件置起
- 无效数据从移位寄存器传送到 USART_DR 寄存器。
- 在单字节通信时，没有中断产生。然而，这个位和 RXNE 位同时置起，后者将产生中断。在多缓冲器通信情况下，如果 USART_CR3 寄存器中 EIE 位被置位的话，将产生中断。

顺序执行对 USART_SR 和 USART_DR 寄存器的读操作，可复位 FE 位。

接收期间的可配置的停止位

被接收的停止位的个数可以通过控制寄存器 2 的控制位来配置，在正常模式时，可以是 1 或 2 个，在智能卡模式里可能是 0.5 或 1.5 个。

1. 0.5 个停止位(智能卡模式中的接收)：不对 0.5 个停止位进行采样。因此，如果选择 0.5 个停止位则不能检测帧错误和断开帧。
2. 1 个停止位：对 1 个停止位的采样在第 8，第 9 和第 10 采样点上进行。
3. 1.5 个停止位(智能卡模式)：当以智能卡模式发送时，器件必须检查数据是否被正确的发送出去。所以接收器功能块必须被激活(USART_CR1 寄存器中的 RE =1)，并且在停止位的发送期间采样数据线上的信号。如果出现校验错误，智能卡会在发送方采样 NACK 信号时，即总线上停止位对应的时间内时，拉低数据线，以此表示出现了帧错误。FE 在 1.5 个停止位结束时和 RXNE 一起被置起。对 1.5 个停止位的采样是在第 16，第 17 和第 18 采样点进行的。1.5 个的停止位可以被分成 2 部分：一个是 0.5 个时钟周期，期间不做任何事情。随后是 1 个时钟周期的停止位，在这段时间的中点处采样。详见第 22.3.11 节：智能卡。
4. 2 个停止位：对 2 个停止位的采样是在第一停止位的第 8，第 9 和第 10 个采样点完成的。如果第一个停止位期间检测到一个帧错误，帧错误标志将被设置。第二个停止位不再检查帧错误。在第一个停止位结束时 RXNE 标志将被设置。

21.3.4 分数波特率的产生

接收器和发送器的波特率在 USARTDIV 的整数和小数寄存器中的值应设置成相同。

$$\text{Tx / Rx 波特率} = \frac{f_{CK}}{(16 * USARTDIV)}$$

这里的 f_{CK} 是给外设的时钟(PCLK1 用于 USART2、3、4、5，PCLK2 用于 USART1)

USARTDIV 是一个无符号的定点数。这 12 位的值设置在 USART_BRR 寄存器。

注：在写入 USART_BRR 之后，波特率计数器会被波特率寄存器的新值替换。因此，不要在通信进行中改变波特率寄存器的数值。

如何从 USART_BRR 寄存器值得到 USARTDIV

例 1：

如果DIV_Mantissa = 27 ， DIV_Fraction = 12 (USART_BRR=0x1BC),

于是

Mantissa (USARTDIV) = 27

Fraction (USARTDIV) = 12/16 = 0.75

所以USARTDIV = 27.75

例 2：

要求USARTDIV = 25.62,

就有：

DIV_Fraction = 16*0.62 = 9.92

最接近的整数是：10 = 0x0A

DIV_Mantissa = mantissa (25.620) = 25 = 0x19

于是，USART_BRR = 0x19A

例 3：

要求USARTDIV = 50.99

就有：

DIV_Fraction = 16*0.99 = 15.84

最接近的整数是：16 = 0x10 => DIV_frac[3:0]溢出=> 进位必须加到小数部分

DIV_Mantissa = mantissa (50.990 + 进位) = 51 = 0x33

于是：USART_BRR = 0x330，USARTDIV=51

表 82 设置波特率时的误差计算

波特率		f_{PCLK} = 36MHz			f_{PCLK} = 72MHz		
序号	Kbps	实际	置于波特率寄存器中的值	误差%	实际	置于波特率寄存器中的值	误差%
1	2.4	2.400	937.5	0%	2.4	1875	0%
2	9.6	9.600	234.375	0%	9.6	468.75	0%
3	19.2	19.2	117.1875	0%	19.2	234.375	0%
4	57.6	57.6	39.0625	0%	57.6	78.125	0%
5	115.2	115.384	19.5	0.15%	115.2	39.0625	0%
6	230.4	230.769	9.75	0.16%	230.769	19.5	0.16%
7	460.8	461.538	4.875	0.16%	461.538	9.75	0.16%
8	921.6	923.076	2.4375	0.16%	923.076	4.875	0.16%
9	2250	2250	1	0%	2250	2	0%
10	4500	不可能	不可能	不可能	4500	1	0%

注：1. CPU 的时钟频率越低，则某一特定波特率的误差也越低。可以达到的波特率上限可以由这组数据得到。

2. 只有 USART1 使用 PCLK2(最高 72MHz)。其它 USART 使用 PCLK1(最高 36MHz)。

21.3.5 USART 接收器容忍时钟的变化

只有当整体的时钟系统地变化小于 USART 异步接收器能够容忍的范围，USART 异步接收器才能正常地工作。影响这些变化的因素有：

- DTRA：由于发送器误差而产生的变化(包括发送器端振荡器的变化)
- DQUANT：接收器端波特率取整所产生的误差
- DREC：接收器端振荡器的变化
- DTCL：由于传输线路产生的变化(通常是由于收发器在由低变高的转换时序，与由高变低转换时序之间的不一致性所造成)。

需要满足：DTRA + DQUANT + DREC + DTCL < USART 接收器的容忍度

对于正常接收数据，USART 接收器的容忍度等于最大能容忍的变化，它依赖于下述选择：

- 由 USART_CR1 寄存器的 M 位定义的 10 或 11 位字符长度
- 是否使用分数波特率产生

表 83 当 DIV_Fraction=0 时，USART 接收器的容忍度

M 位	认为 NF 是错误	不认为 NF 是错误
0	3.75%	4.375%
1	3.41%	3.97%

表 84 当 DIV_Fraction!=0 时，USART 接收器的容忍度

M 位	认为 NF 是错误	不认为 NF 是错误
0	3.33%	3.88%
1	3.03%	3.53%

注：在特殊的情况下，即当收到的帧包含一些在 M=0 时，正好是 10 位(M=1 时是 11 位)的空闲帧，上面 2 个表格中的数据可能会有少许出入。

21.3.6 多处理器通信

通过 USART 可以实现多处理器通信(将几个 USART 连在一个网络里)。例如某个 USART 设备可以 是主，它的 TX 输出和其他 USART 从设备的 RX 输入相连接；USART 从设备各自的 TX 输出逻辑地 与在一起，并且和主设备的 RX 输入相连接。

在多处理器配置中，我们通常希望只有被寻址的接收者才被激活，来接收随后的数据，这样就可以减少由未被寻址的接收器的参与带来的多余的 USART 服务开销。未被寻址的设备可启用其静默功能置于静默模式。在静默模式里：

- 任何接收状态位都不会被设置。
- 所有接收中断被禁止。
- USART_CR1 寄存器中的 RWU 位被置 1。RWU 可以被硬件自动控制或在某个条件下由软件写入。

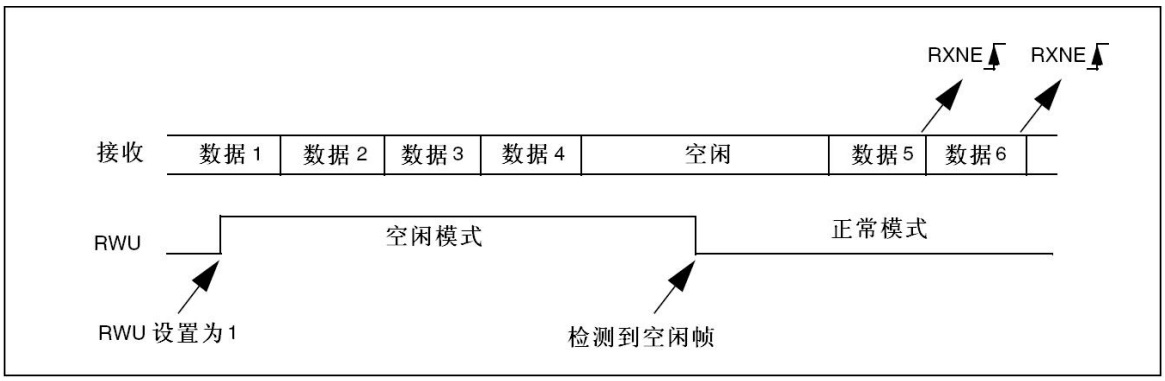
根据 USART_CR1 寄存器中的 WAKE 位状态，USART 可以用二种方法进入或退出静默模式。

- 如果 WAKE 位被复位：进行空闲总线检测。
- 如果 WAKE 位被设置：进行地址标记检测。

空闲总线检测(WAKE=0)

当 RWU 位被写 1 时，USART 进入静默模式。当检测到一空闲帧时，它被唤醒。然后 RWU 被硬件清零，但是 USART_SR 寄存器中的 IDLE 位并不置起。RWU 还可以被软件写 0。下图给出利用空闲总线检测来唤醒和进入静默模式的一个例子

图 179 利用空闲总线检测的静默模式



地址标记(address mark)检测(WAKE=1)

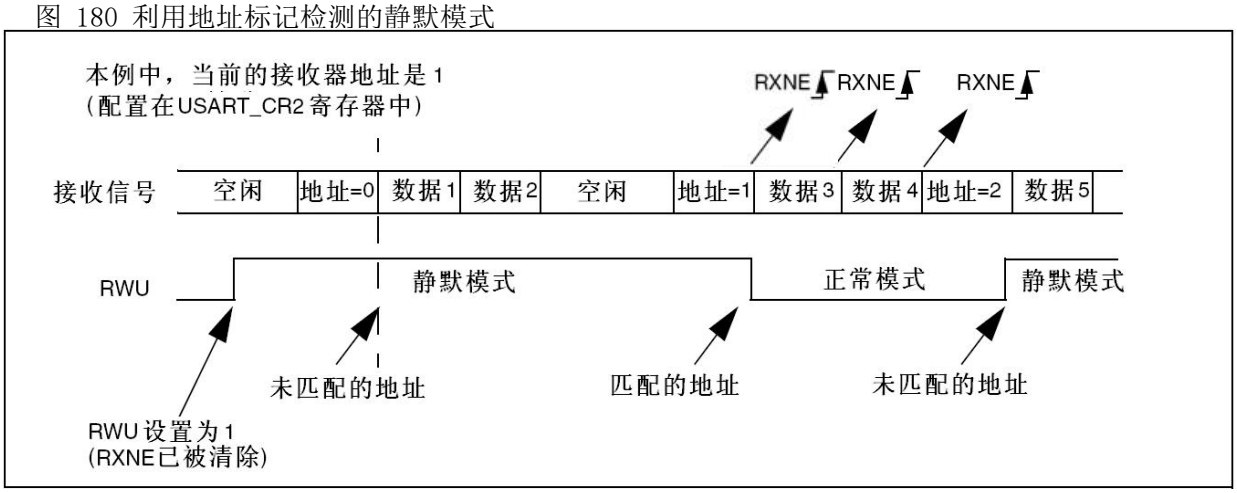
在这个模式里，如果 MSB 是 1，该字节被认为是地址，否则被认为是数据。在一个地址字节中，目标接收器的地址被放在 4 个 LSB 中。这个 4 位地址被接收器同它自己地址做比较，接收器的地址被编程在 USART_CR2 寄存器的 ADD。

如果接收到的字节与它的编程地址不匹配时，USART 进入静默模式。此时，硬件设置 RWU 位。接收该字节既不会设置 RXNE 标志也不会产生中断或发出 DMA 请求，因为 USART 已经在静默模 式。

当接收到的字节与接收器内编程地址匹配时，USART 退出静默模式。然后 RWU 位被清零，随后的字节被正常接收。收到这个匹配的地址字节时将设置 RXNE 位，因为 RWU 位已被清零。

当接收缓冲器不包含数据时(USART_SR 的 RXNE=0)，RWU 位可以被写 0 或 1。否则，该次写操 作被忽略。下图给出利用地址标记检测来唤醒和进入静默模式的例子。

图 180 利用地址标记检测的静默模式



21.3.7 校验控制

设置 USART_CR1 寄存器上的 PCE 位，可以使能奇偶控制(发送时生成一个奇偶位，接收时进行奇偶校验)。根据 M 位定义的帧长度，可能的 USART 帧格式列在下表中。

表 85 帧格式

M 位	PCE 位	USART 帧
0	0	| 起始位| 8 位数据| 停止位|
0	1	| 起始位| 7 位数据| 奇偶检验位| 停止位|
1	0	| 起始位| 9 位数据| 停止位|
1	1	| 起始位| 8 位数据| 奇偶检验位| 停止位|

注意：在用地址标记唤醒设备时，地址的匹配只考虑到数据的 MSB 位，而不用关心校验位。(MSB 是数据位中最后发出的，后面紧跟校验位或者停止位)

偶校验：校验位使得一帧中的 7 或 8 个 LSB 数据以及校验位中'1'的个数为偶数。

例如：数据=00110101，有 4 个'1'，如果选择偶校验(在 USART_CR1 中的 PS＝0)，校验位将是'0'。

奇校验：此校验位使得一帧中的 7 或 8 个 LSB 数据以及校验位中'1'的个数为奇数。

例如：数据=00110101，有 4 个'1'，如果选择奇校验(在 USART_CR1 中的 PS＝1)，校验位将是'1'。

传输模式：如果 USART_CR1 的 PCE 位被置位，写进数据寄存器的数据的 MSB 位被校验位替换后发送出去(如果选择偶校验偶数个'1'，如果选择奇校验奇数个'1')。如果奇偶校验失败，USART_SR 寄存器中的 PE 标志被置'1'，并且如果 USART_CR1 寄存器的 PEIE 在被预先设置的话，中断产生。

21.3.8 LIN(局域互联网)模式

LIN 模式是通过设置 USART_CR2 寄存器的 LINEN 位选择。在 LIN 模式下，下列位必须保持为 0：

- USART_CR2 寄存器的 CLKEN 位
- USART_CR3 寄存器的 STOP[1:0]，SCEN，HDSEL 和 IREN

LIN 发送

22.3.2 节里所描述的同样步骤适用于 LIN 主发送，但和正常 USART 发送有以下区别：

- 清零 M 位以配置 8 位字长
- 置位 LINEN 位以进入 LIN 模式。这时，置位 SBK 将发送 13 位'0'作为断开符号。然后发一位'1'，以允许对下一个开始位的检测。

LIN 接收

当 LIN 模式被使能时，断开符号检测电路被激活。该检测完全独立于 USART 接收器。断开符号只要一出现就能检测到，不管是在总线空闲时还是在发送某数据帧其间，数据帧还未完成，

当接收器被激活时(USART_CR1 的 RE=1)，电路监测 RX 上的起始信号。监测起始位的方法同检测断开符号或数据是一样的。当起始位被检测到后，电路对每个接下来的位，在每个位的第 8，9，10 个过采样时钟点上进行采样。如果 10 个(当USART_CR2 的LBDL = 0) 或11 个(当USART_CR2 的 LBDL = 1)连续位都是'0'，并且又跟着一个定界符，USART_SR 的 LBD 标志被设置。如果 LBDIE 位=1，中断产生。在确认断开符号前，要检查定界符，因为它意味 RX 线已经 回到高电平。

如果在第 10 或 11 个采样点之前采样到了'1'，检测电路取消当前检测并重新寻找起始位。如果 LIN模式被禁止，接收器继续如正常 USART 那样工作，不需要考虑检测断开符号。

如果 LIN 模式没有被激活(LINEN=0)，接收器仍然正常工作于 USART 模式，不会进行断开检测。

如果 LIN 模式被激活(LINEN=1)，只要一发生帧错误(也就是停止位检测到'0'，这种情况出现在断开帧)，接收器就停止，直到断开符号检测电路接收到一个'1'(这种情况发生于断开符号没有完整的发出来)，或一个定界符(这种情况发生于已经检测到一个完整的断开符号)。

图 181 说明了断开符号检测器状态机的行为和断开符号标志的关系。 图 182 给出了一个断开帧的例子。

图 181 LIN 模式下的断开检测(11 位断开长度– 设置了 LBDL 位)

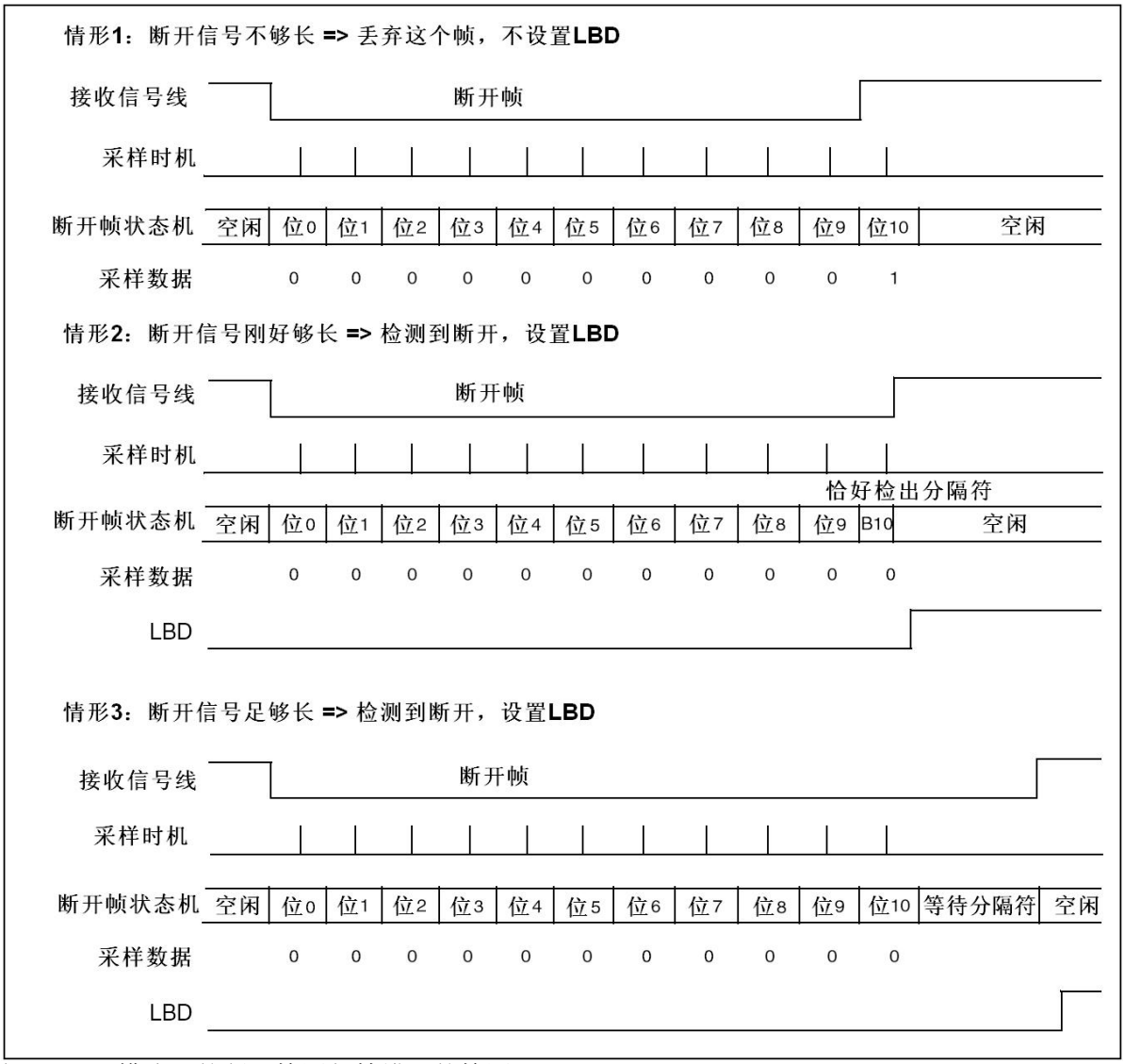
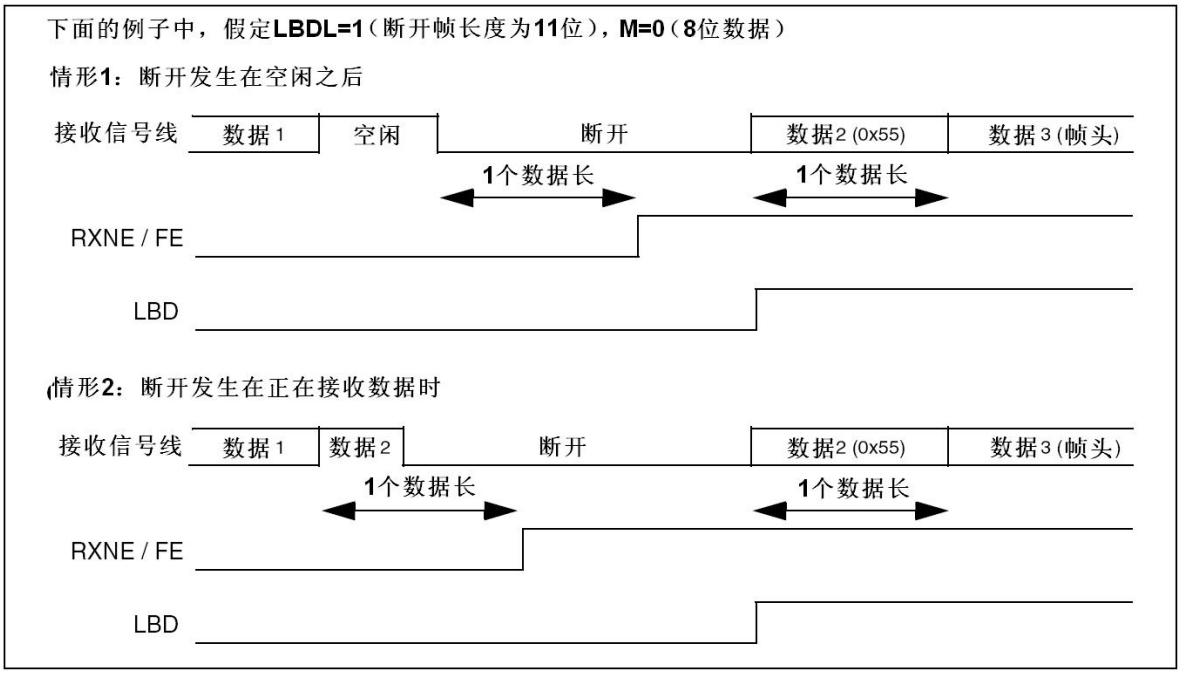


图 182 LIN 模式下的断开检测与帧错误的检测



21.3.9 USART 同步模式

通过在 USART_CR2 寄存器上写 CLKEN 位选择同步模式

在同步模式里，下列位必须保持清零状态：

- USART_CR2 寄存器中的 LINEN 位
- USART_CR3 寄存器中的 SCEN,HDSEL 和 IREN 位

USART 允许用户以主模式方式控制双向同步串行通信。CK 脚是 USART 发送器时钟的输出。在起始位和停止位期间，CK 脚上没有时钟脉冲。根据 USART_CR2 寄存器中 LBCL 位的状态，决定 在最后一个有效数据位期间产生或不产生时钟脉冲。USART_CR2 寄存器的 CPOL 位允许用户选择时钟极性，USART_CR2 寄存器上的 CPHA 位允许用户选择外部时钟的相位(见图 183、图 184 和图 185)。

在总线空闲期间，实际数据到来之前以及发送断开符号的时候，外部 CK 时钟不被激活。

同步模式时，USART 发送器和异步模式里工作一模一样。但是因为 CK 是与 TX 同步的(根据 CPOL 和 CPHA)，所以 TX 上的数据是随 CK 同步发出的。

同步模式的 USART 接收器工作方式与异步模式不同。如果 RE=1，数据在 CK 上采样(根据 CPOL 和 CPHA 决定在上升沿还是下降沿)，不需要任何的过采样。但必须考虑建立时间和持续时间(取决于波特率，1/16 位时间)。

注意： *1．CK 脚同 TX 脚一起联合工作。因而，只有在使能了发送器(TE＝1)，并且发送数据时(写入数据至 USART_DR 寄存器)才提供时钟。这意味着在没有发送数据时是不可能接收一个同步数据的。*

2．LBCL,CPOL 和 CPHA 位的正确配置，应该在发送器和接收器都被禁止时；当使能了发送器或接收器时，这些位不能被改变

3．建议在同一条指令中设置 TE 和 RE，以减少接收器的建立时间和保持时间。

4． USART 只支持主模式：它不能用来自其他设备的输入时钟接收或发送数据(CK 永远是输出)。

图 183 USART 同步传输的例子

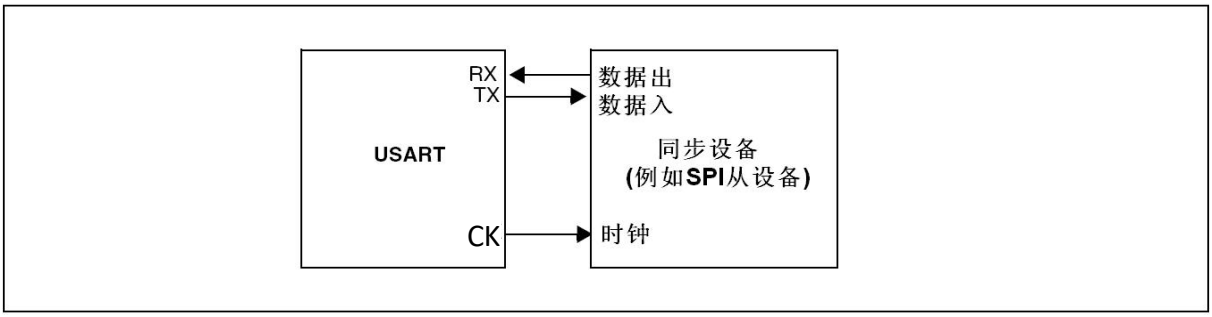


图 184 USART 数据时钟时序示例(M=0)

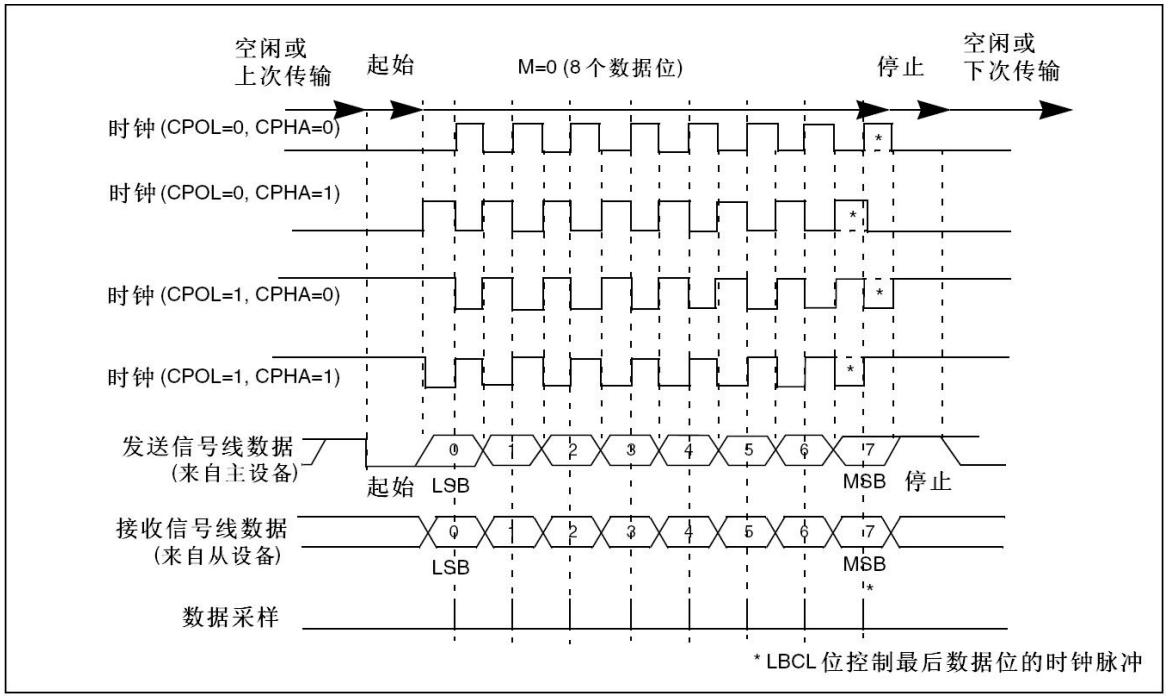


图 185 USART 数据时钟时序示例(M=1)

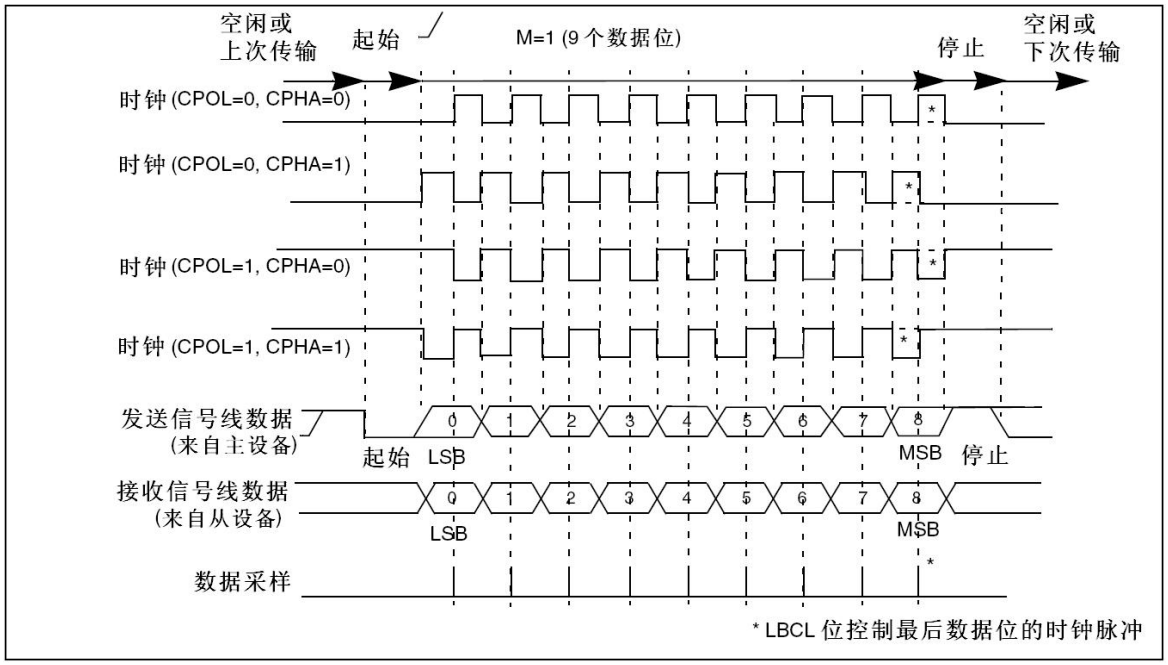
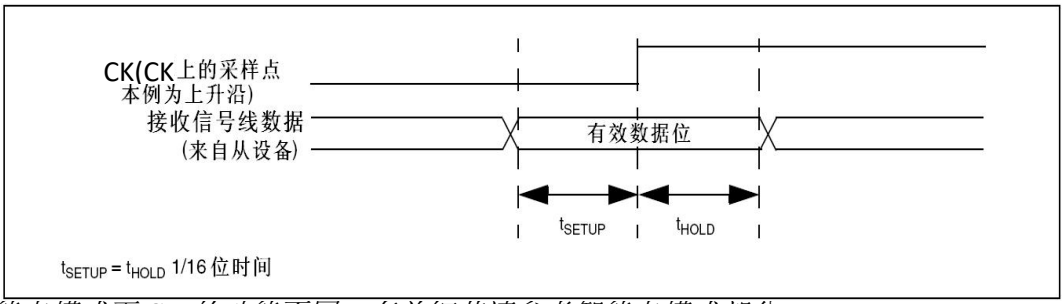


图 186 RX 数据采样/保持时间



注：在智能卡模式下 CK 的功能不同，有关细节请参考智能卡模式部分。

21.3.10 单线半双工通信

单线半双方模式通过设置 USART_CR3 寄存器的 HDSEL 位选择。在这个模式里，下面的位必须保持清零状态：

- USART_CR2 寄存器的 LINEN 和 CLKEN 位
- USART_CR3 寄存器的 SCEN 和 IREN 位

USART 可以配置成遵循单线半双工协议。在单线半双工模式下，TX 和 RX 引脚在芯片内部互连。使用控制位"HALF DUPLEX SEL"(USART_CR3 中的 HDSEL 位)选择半双工和全双工通信。当 HDSEL 为'1'时

- RX 不再被使用
- 当没有数据传输时，TX 总是被释放。因此，它在空闲状态的或接收状态时表现为一个标准 I/O 口。这就意味该 I/O 在不被 USART 驱动时，必须配置成悬空输入(或开漏的输出高)。

除此以外，通信与正常 USART 模式类似。由软件来管理线上的冲突(例如通过使用一个中央仲裁器)。特别的是，发送从不会被硬件所阻碍。当 TE 位被设置时，只要数据一写到数据寄存器上，发送就继续。

21.3.11 智能卡

设置 USART_CR3 寄存器的 SCEN 位选择智能卡模式。在智能卡模式下，下列位必须保持清零：

- USART_CR2 寄存器的 LINEN 位
- USART_CR3 寄存器的 HDSEL 位和 IREN 位

此外，CLKEN 位可以被设置，以提供时钟给智能卡。

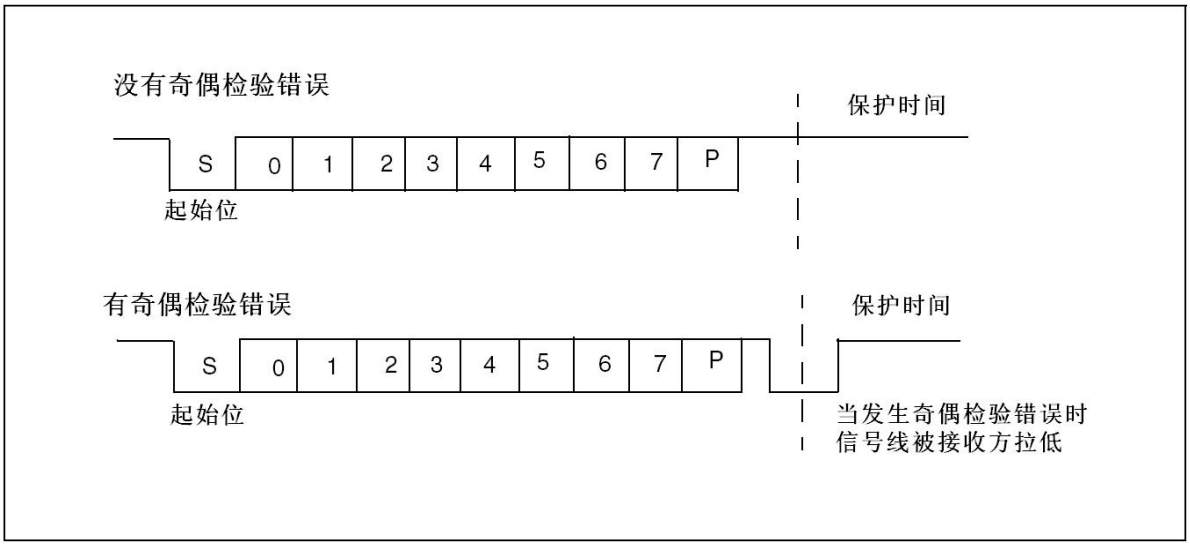
该接口符合 ISO7816-3 标准，支持智能卡异步协议。USART 应该被设置为：

- 8 位数据位加校验位：此时 USART_CR1 寄存器中 M=1、PCE=1
- 发送和接收时为 1.5 个停止位：即 USART_CR2 寄存器的 STOP=11

注： *也可以在接收时选择 0.5 个停止位，但为了避免在 2 种配置间转换，建议在发送和接收时使用 1.5 个停止位。*

下图给出的例子说明了数据线上，在有校验错误和没校验错误两种情况下的信号。

图 187 ISO7816-3 异步协议



当与智能卡相连接时，USART 的 TX 驱动一根智能卡也驱动的双向线。所以 TX 必须配置成开漏。

智能卡是一个单线半双工通信协议

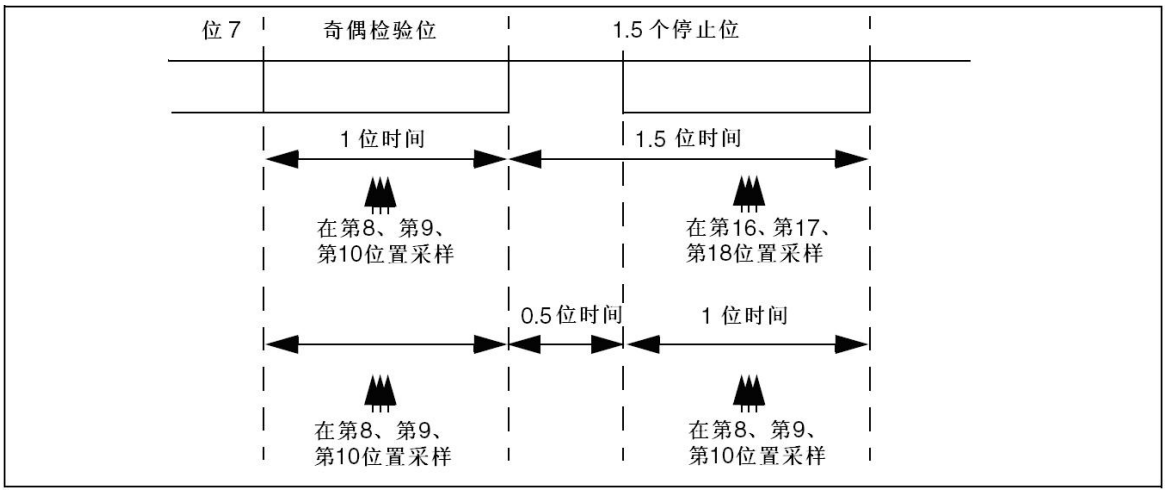
- 从发送移位寄存器把数据发送出去，要被延时最小 1/2 波特时钟。在正常操作时，一个满的发送移位寄存器将在下一个波特时钟沿开始向外移出数据。在智能卡模式里，此发送被延迟 1/2 波特时钟。
- 如果在接收一个设置为 0.5 或 1.5 个停止位的数据帧期间，检测到一奇偶校验错误，在完成接收该帧后(即停止位结束时)，发送线被拉低一个波特时钟周期。这是告诉智能卡发送到 USART 的数据没有被正确地接收到。此 NACK 信号(拉低发送线一个波特时钟周期)在发送端将产生一个帧错误(发送端被配置成 1.5 个停止位)。应用程序可以根据协议处理重新发送数据。如果设置了 NACK 控制位，发生校验错误时接收器会给出一个 NACK 信号；否则就不会 发送 NACK。
- TC 标志的置起可以通过编程保护时间寄存器得以延时。在正常操作时，当发送移位寄存器变空并且没有新的发送请求出现时，TC 被置起。在智能卡模式里，空的发送移位寄存器将触发保护时间计数器开始向上计数，直到保护时间寄存器中的值。TC 在这段时间被强制拉低。当保护时间计数器达到保护时间寄存器中的值时，TC 被置高。
- TC 标志的撤销不受智能卡模式的影响。
- 如果发送器检测到一个帧错误(收到接收器的 NACK 信号)，发送器的接收功能模块不会把 NACK 当作起始位检测。根据 ISO 协议，接收到的 NACK 的持续时间可以是 1 或 2 波特时钟周期。
- 在接收器这边，如果一个校验错误被检测到，并且 NACK 被发送，接收器不会把 NACK 检测成起始位。

注意： *1. 断开符号在智能卡模式里没有意义。一个带帧错误的 00h 数据将被当成数据而不是断开符号。*

2. 当来回切换 TE 位时，没有 IDLE 帧被发送。ISO 协议没有定义 IDLE 帧。

下图详述了 USART 是如何采样 NACK 信号的。在这个例子里，USART 正在发送数据，并且被配置成 1.5 个停止位。为了检查数据的完整性和 NACK 信号，USART 的接收功能块被激活。

图 188 使用 1.5 停止位检测奇偶检验错



USART 可以通过 CK 输出为智能卡提供时钟。在智能卡模式里，CK 不和通信直接关联，而是先通过一个 5 位预分频器简单地用内部的外设输入时钟来驱动智能卡的时钟。分频率在预分频寄存器 USART_GTPR 中配置。CK 频率可以从 $f_{CK}/2$ 到 $f_{CK}/62$，这里的 f_{CK} 是外设输入时钟。

21.3.12 IrDA SIR ENDEC 功能模块

通过设置 USART_CR3 寄存器的 IREN 位选择 IrDA 模式。在 IRDA 模式里，下列位必须保持清零：

- USART_CR2 寄存器的 LINEN,STOP 和 CLKEN 位
- USART_CR3 寄存器的 SCEN 和 HDSEL 位。

IrDA SIR 物理层规定使用反相归零调制方案(RZI)，该方案用一个红外光脉冲代表逻辑'0'(见图

189)。SIR 发送编码器对从 USART 输出的 NRZ(非归零)比特流进行调制。输出脉冲流被传送到一个外部输出驱动器和红外 LED。USART 为 SIR ENDEC 最高只支持到 115.2Kbps 速率。在正常模式里，脉冲宽度规定为一个位周期的 3/16。

SIR 接收解码器对来自红外接收器的归零位比特流进行解调，并将接收到的 NRZ 串行比特流输出到 USART。在空闲状态里，解码器输入通常是高(标记状态 marking state)。发送编码器输出的极性和解码器的输入相反。当解码器输入低时，检测到一个起始位。

- IrDA 是一个半双工通信协议。如果发送器忙(也就是 USART 正在送数据给 IrDA 编码器)， IrDA 接收线上的任何数据将被 IrDA 解码器忽视。如果接收器忙(也就是 USART 正在接收从 IrDA 解码器来的解码数据)，从 USART 到 IrDA 的 TX 上的数据将不会被 IrDA 编码。当接收数据时，应该避免发送，因为将被发送的数据可能被破坏。
- SIR 发送逻辑把'0'作为高脉冲发送，把'1'作为低电平发送。脉冲的宽度规定为正常模式时位周期的 3/16(见图 190)。
- SIR 接收逻辑把高电平状态解释为'1'，把低脉冲解释为'0'。
- 发送编码器输出与解码器输入有着相反的极性。当空闲时，SIR 输出处于低状态。
- SIR 解码器把 IrDA 兼容的接收信号转变成给 USART 的比特流。
- IrDA 规范要求脉冲要宽于 1.41us。脉冲宽度是可编程的。接收器端的尖峰脉冲检测逻辑滤除宽度小于 2 个 PSC 周期的脉冲(PSC 是在 IrDA 低功耗波特率寄存器 USART_GTPR 中编程的预分频值)。宽度小于 1 个 PSC 周期的脉冲一定被滤除掉，但是那些宽度大于 1 个而小于 2 个 PSC 周期的脉冲可能被接收或滤除，那些宽度大于 2 个周期的将被视为一个有效的脉冲。当 PSC=0 时，IrDA 编码器/解码器不工作。
- 接收器可以与一低功耗发送器通信。
- 在 IrDA 模式里，USART_CR2 寄存器上的 STOP 位必须配置成 1 个停止位。

IrDA 低功耗模式

发送器

在低功耗模式，脉冲宽度不再持续 3/16 个位周期。取而代之，脉冲的宽度是低功耗波特率的 3倍，它最小可以是 1.42MHz。通常这个值是 1.8432MHz(1.42 MHz < PSC < 2.12 MHz)。一个低功耗模式可编程分频器把系统时钟进行分频以达到这个值。

接收器

低功耗模式的接收类似于正常模式的接收。为了滤除尖峰干扰脉冲，USART 应该滤除宽度短于 1 个 PSC 的脉冲。只有持续时间大于 2 个周期的 IrDA 低功耗波特率时钟(USART_GTPR 中的 PSC)的低电平信号才被接受为有效的信号。

注意：
1. *宽度小于 2 个大于 1 个 PSC 周期的脉冲可能会也可能不会被滤除。*
2. *接收器的建立时间应该由软件管理。IrDA 物理层技术规范规定了在发送和接收之间最小要有 10ms 的延时(IrDA 是一个半双工协议)*

图 189 IrDA SIR ENDEC – 框图

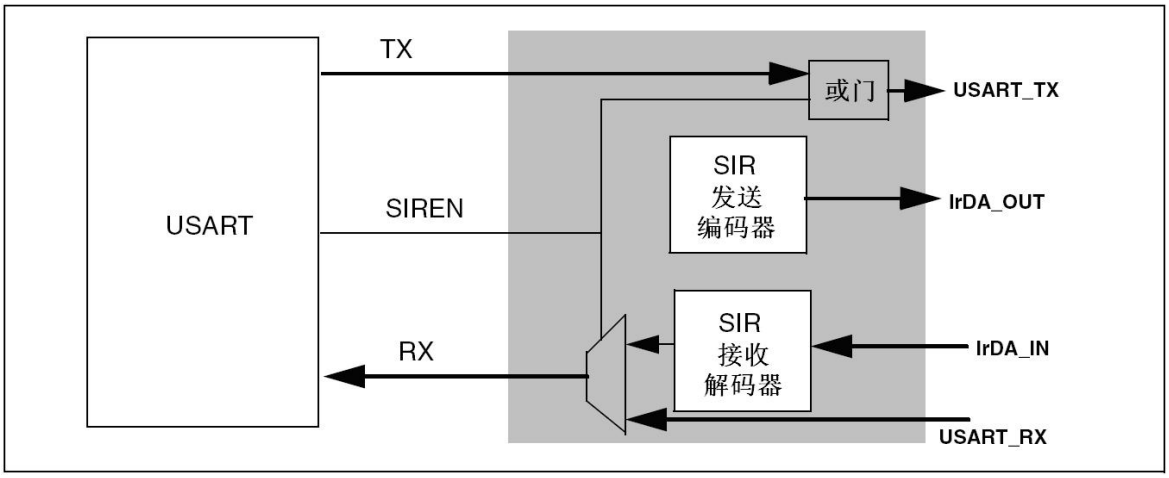
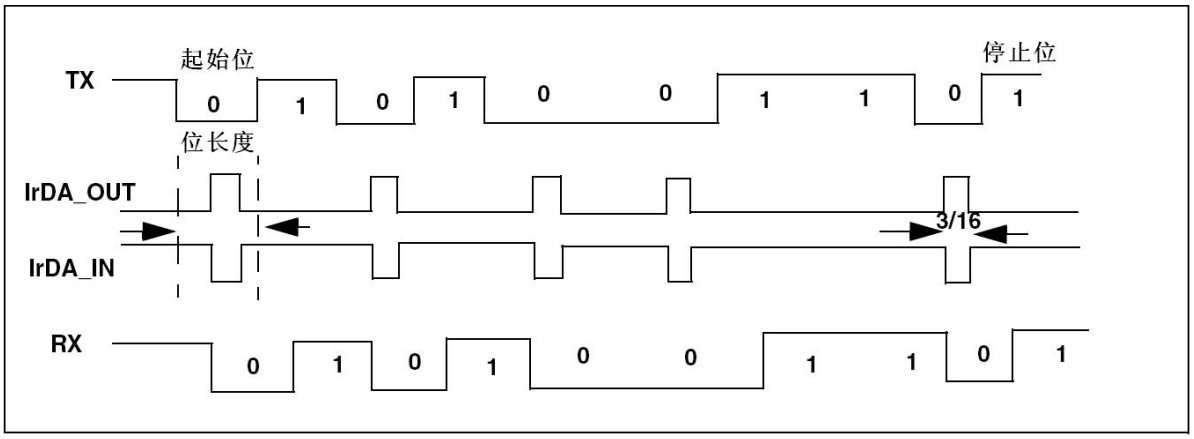


图 190 IrDA 数据调制(3/16) – 普通模式



21.3.13 利用 DMA 连续通信

USART 可以利用 DMA 连续通信。Rx 缓冲器和 Tx 缓冲器的 DMA 请求是分别产生的。

注意：参考产品技术说明以确定是否可用 DMA 控制器。如果所用产品无 DMA 功能，应按 22.3.2 节或 22.3.3 节里所描述的方法使用 USART。在 USART2_SR 寄存器里，可以清零 TXE/RXNE 标志来实现连续通信。

利用 DMA 发送

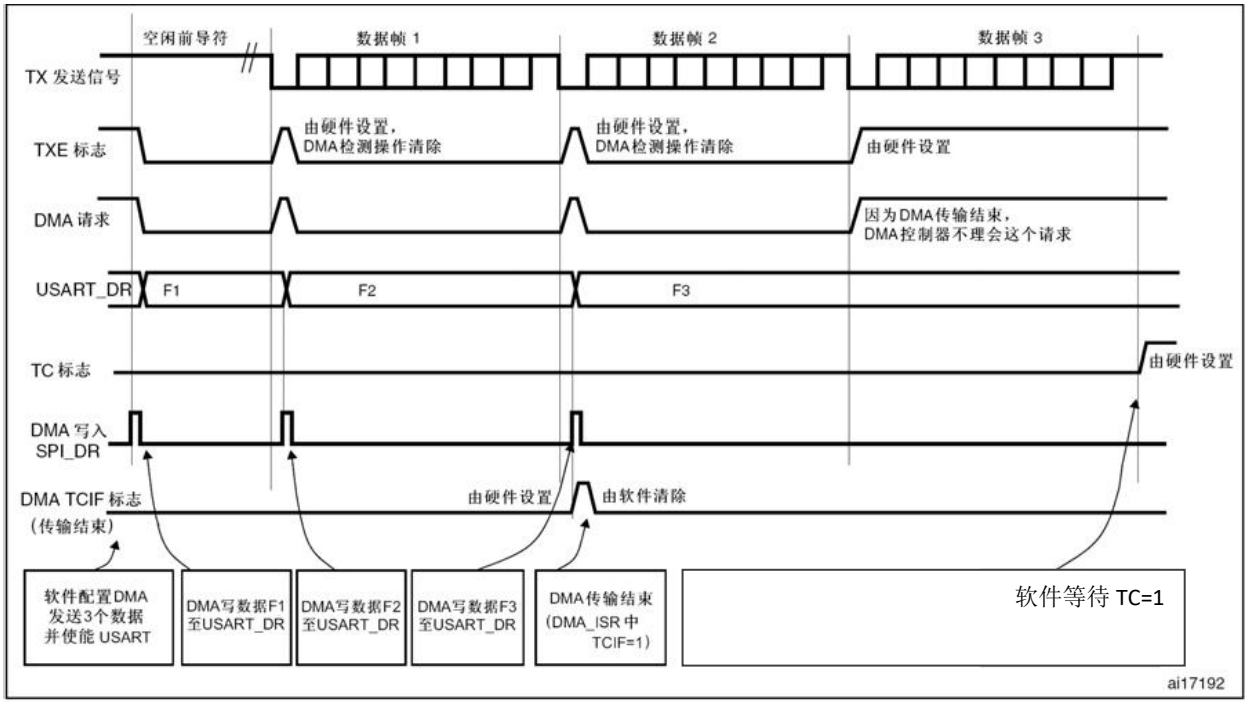
使用 DMA 进行发送，可以通过设置 USART_CR3 寄存器上的 DMAT 位激活。当 TXE 位被置为'1' 时，DMA 就从指定的 SRAM 区传送数据到 USART_DR 寄存器。为 USART 的发送分配一个 DMA 通道的步骤如下(x 表示通道号)：

1. 在 DMA 控制寄存器上将 USART_DR 寄存器地址配置成 DMA 传输的目的地址。在每个 TXE 事件后，数据将被传送到这个地址。
2. 在 DMA 控制寄存器上将存储器地址配置成 DMA 传输的源地址。在每个 TXE 事件后，将从 此存储器区读出数据并传送到 USART_DR 寄存器。
3. 在 DMA 控制寄存器中配置要传输的总的字节数。
4. 在 DMA 寄存器上配置通道优先级。
5. 根据应用程序的要求，配置在传输完成一半还是全部完成时产生 DMA 中断。
6. 在 DMA 寄存器上激活该通道。 当传输完成 DMA 控制器指定的数据量时，DMA 控制器在该 DMA 通道的中断向量上产生一中断。

在发送模式下，当 DMA 传输完所有要发送的数据时，DMA 控制器设置 DMA_ISR 寄存器的 TCIF

标志；监视 USART_SR 寄存器的 TC 标志可以确认USART通信是否结束，这样可以在关闭USART 或进入停机模式之前避免破坏最后一次传输的数据；软件需要先等待 TXE=1，再等待TC=1。

图 191 利用 DMA 发送



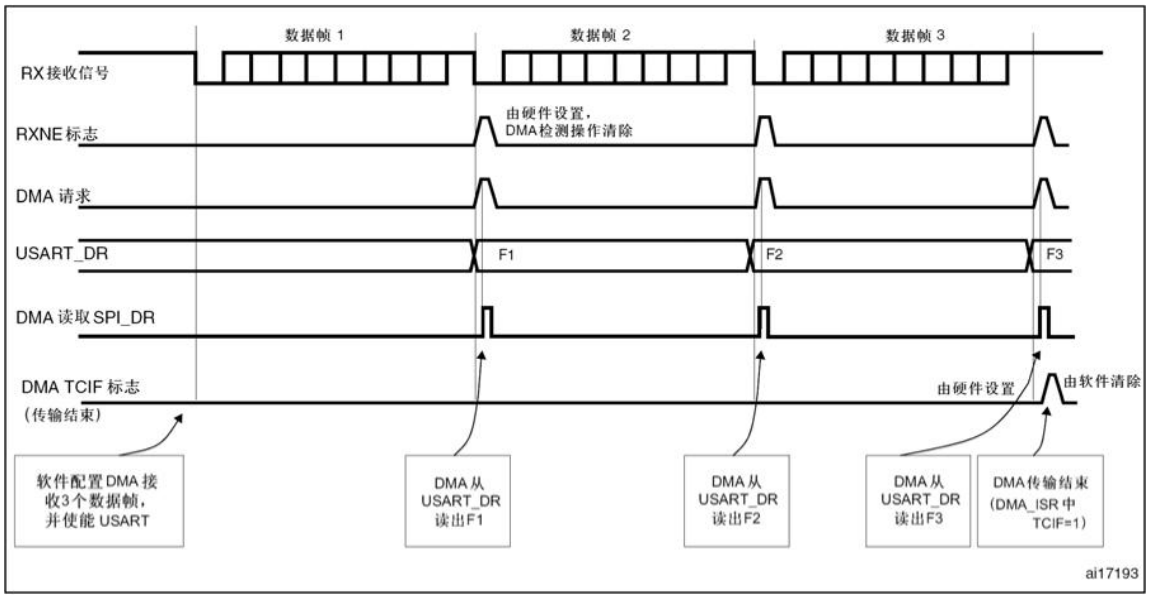
利用 DMA 接收

可以通过设置 USART_CR3 寄存器的 DMAR 位激活使用 DMA 进行接收，每次接收到一个字节，DMA 控制器就就把数据从 USART_DR 寄存器传送到指定的 SRAM 区(参考 DMA 相关说明)。为
USART 的接收分配一个 DMA 通道的步骤如下(x 表示通道号)：

1. 通过 DMA 控制寄存器把 USART_DR 寄存器地址配置成传输的源地址。在每个 RXNE 事件后，将从此地址读出数据并传输到存储器。
2. 通过 DMA 控制寄存器把存储器地址配置成传输的目的地址。在每个 RXNE 事件后，数据将从 USART_DR 传输到此存储器区。
3. 在 DMA 控制寄存器中配置要传输的总的字节数。
4. 在 DMA 寄存器上配置通道优先级。。
5. 根据应用程序的要求配置在传输完成一半还是全部完成时产生 DMA 中断。
6. 在 DMA 控制寄存器上激活该通道。

当接收完成 DMA 控制器指定的传输量时，DMA 控制器在该 DMA 通道的中断矢量上产生一中断。

图 192 利用 DMA 接收



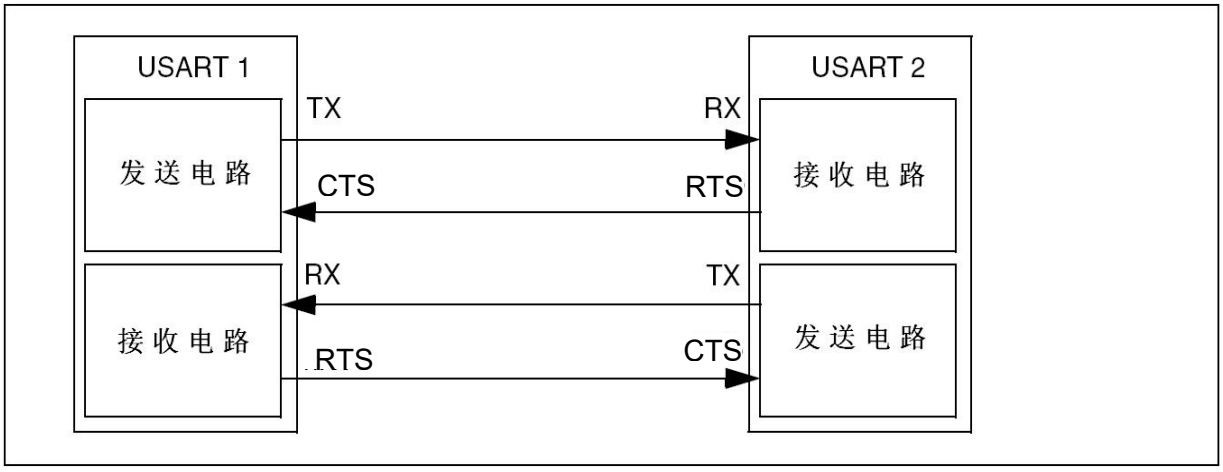
多缓冲器通信中的错误标志和中断产生

在多缓冲器通信的情况下，通信期间如果发生任何错误，在当前字节传输后将置起错误标志。如果中断使能位被设置，将产生中断。在单个字节接收的情况下，和 RXNE 一起被置起的帧错误、溢出错误和噪音标志，有单独的错误标志中断使能位；如果设置了，会在当前字节传输结束后，产生中断。

21.3.14 硬件流控制

利用 CTS 输入和 RTS 输出可以控制 2 个设备间的串行数据流。下图表明在这个模式里如何连接 2 个设备。

图 193 两个 USART 间的硬件流控制

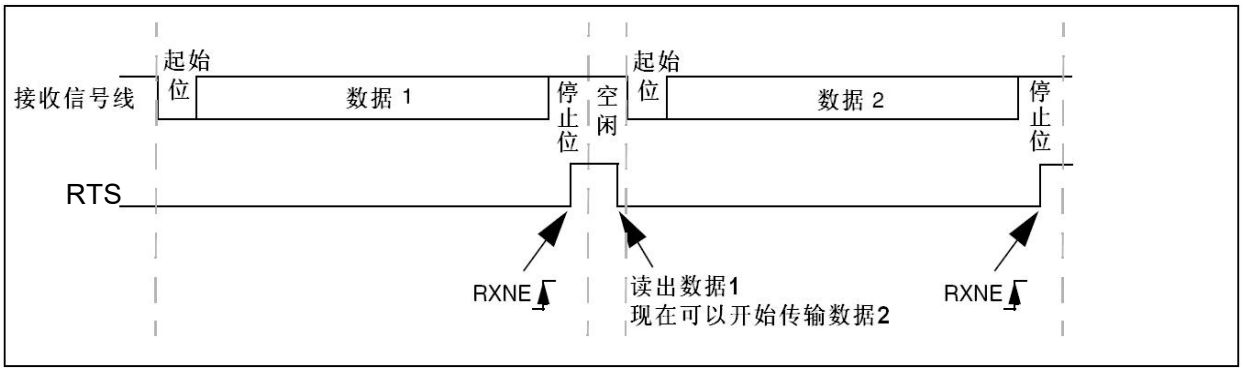


通过将 UASRT_CR3 中的 RTSE 和 CTSE 置位，可以分别独立地使能 RTS 和 CTS 流控制。

RTS 流控制

如果 RTS 流控制被使能(RTSE=1)，只要 USART 接收器准备好接收新的数据，RTS 就变成有效(接低电平)。当接收寄存器内有数据到达时，RTS 被释放，由此表明希望在当前帧结束时停止数据传输。下图是一个启用 RTS 流控制的通信的例子。

图 194 RTS 流控制

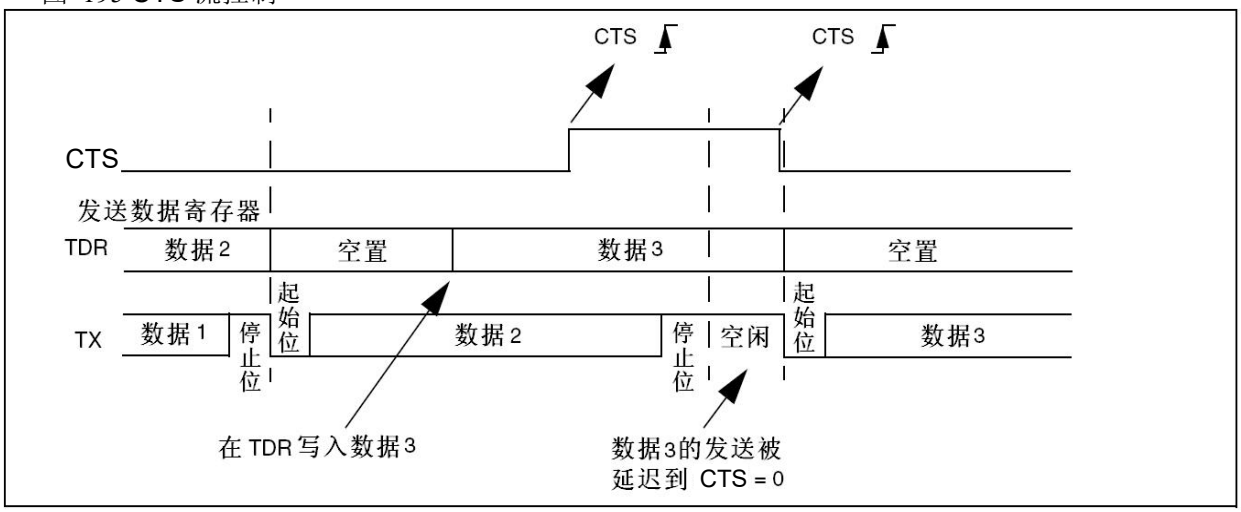


CTS 流控制

如果 CTS 流控制被使能(CTSE=1)，发送器在发送下一帧前检查 CTS 输入。如果 CTS 有效(被拉成低电平)，则下一个数据被发送(假设那个数据是准备发送的，也就是 TXE=0)，否则下一帧数据不被发出去。若 CTS 在传输期间被变成无效，当前的传输完成后停止发送。

当 CTSE=1 时，只要 CTS 输入一变换状态，硬件就自动设置 CTSIF 状态位。它表明接收器是否准备好进行通信。如果设置了 USART_CT3 寄存器的 CTSIE 位，则产生中断。下图是一个启用CTS 流控制通信的例子。

图 195 CTS 流控制



21.4 USART 中断请求

表 86 USART 中断请求

中断事件	事件标志	使能位
发送数据寄存器空	TXE	TXEIE
CTS 标志	CTS	CTSIE
发送完成	TC	TCIE
接收数据就绪可读	TXNE	TXNEIE
检测到数据溢出	ORE	
检测到空闲线路	IDLE	IDLEIE
奇偶检验错	PE	PEIE
断开标志	LBD	LBDIE
噪声标志，多缓冲通信中的溢出错误和帧错误	NE 或 ORT 或 FE	EIE(1)

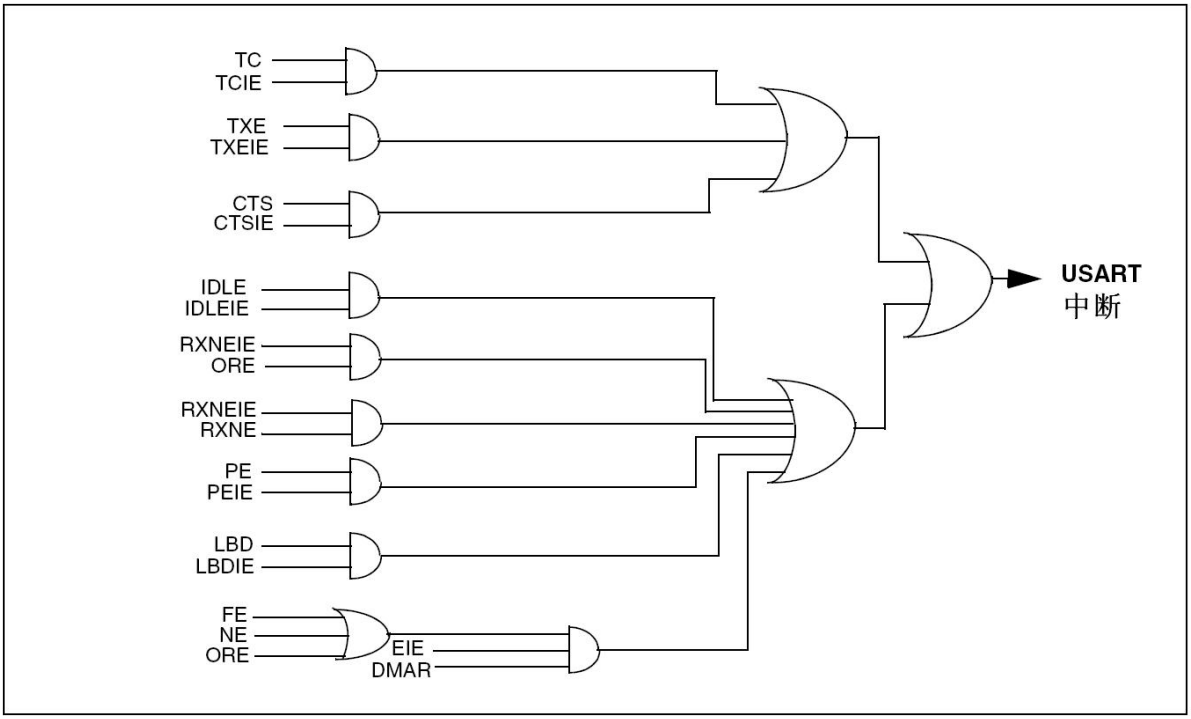
1. 仅当使用 DMA 接收数据时，才使用这个标志位。

USART 的各种中断事件被连接到同一个中断向量(见下图)，有以下各种中断事件：

- 发送期间：发送完成、清除发送、发送数据寄存器空。
- 接收期间：空闲总线检测、溢出错误、接收数据寄存器非空、校验错误、LIN 断开符号检 测、噪音标志(仅在多缓冲器通信)和帧错误(仅在多缓冲器通信)。

如果设置了对应的使能控制位，这些事件就可以产生各自的中断。

图 196 USART 中断映像图



21.5 USART 模式配置

表 87 USART 模式设置

USART 模式	USART1	USART2	USART3
异步模式	◎	◎	◎
硬件流控制	◎	◎	◎
多缓存通讯(DMA)	◎	◎	◎
多处理器通讯	◎	◎	◎
同步	◎	◎	◎
智能卡	◎	◎	◎
半双工(单线模式)	◎	◎	◎
IrDA	◎	◎	◎
LIN	◎	◎	◎

(1) ◎ = 支持，● = 不支持该应用

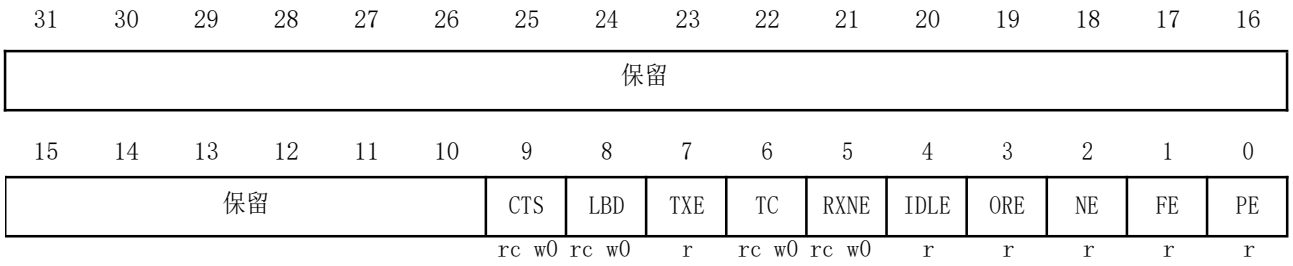
21.6 USART 寄存器描述

有关寄存器描述里所使用的缩写，请参考第 2.1 节。 可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

21.6.1 状态寄存器(USART_SR)

地址偏移：0x00

复位值：0x00C0



位 31:10	保留位，硬件强制为 0
位 9	**CTS**: CTS 标志(CTS flag) 如果设置了 CTSE 位，当 CTS 输入变化状态时，该位被硬件置高。由软件将其清零。 如果 USART_CR3 中的 CTSIE 为'1'，则产生中断。 0：CTS 状态线上没有变化； 1：CTS 状态线上发生变化。
位 8	**LBD**: LIN 断开检测标志(LIN break detection flag) 当探测到 LIN 断开时，该位由硬件置'1'， 由软件清'0'(向该位写 0)。如果 USART_CR3 中的 LBDIE = 1，则产生中断。 0：没有检测到 LIN 断开； 1：检测到 LIN 断开。 注意：若 LBDIE=1，当 LBD 为'1'时要产生中断。
位 7	**TXE**:发送数据寄存器空(Transmit data register empty) 当 TDR 寄存器中的数据被硬件转移到移位寄存器的时候，该位被硬件置位。如果 USART_CR1 寄存器中的 TXEIE 为 1，则产生中断。对 USART_DR 的写操作，将该位清零。 0：数据还没有被转移到移位寄存器； 1：数据已经被转移到移位寄存器。 注意：单缓冲器传输中使用该位。
位 6	**TC**：发送完成(Transmission complete) 当包含有数据的一帧发送完成后，并且 TXE=1 时，由硬件将该位置'1'。如果 USART_CR1 中的 TCIE 为'1'，则产生中断。由软件序列清除该位(先读 USART_SR，然后写入 USART_DR)。TC 位也可以通过写入'0'来清除，只有在多缓存通讯中才推荐这种清除程序。 0：发送还未完成； 1：发送完成。
位 5	**RXNE**：读数据寄存器非空(Read data register not empty) 当 RDR 移位寄存器中的数据被转移到 USART_DR 寄存器中，该位被硬件置位。如果 USART_CR1 寄存器中的 RXNEIE 为 1，则产生中断。对 USART_DR 的读操作可以将该位清零。RXNE 位也可以通过写入 0 来清除，只有在多缓存通讯中才推荐这种清除程序。 0：数据没有收到； 1：收到数据，可以读出。

位 4	**IDLE**：监测到总线空闲(IDLE line detected) 当检测到总线空闲时，该位被硬件置位。如果 USART_CR1 中的 IDLEIE 为'1'，则产生中断。由软件序列清除该位(先读 USART_SR，然后读 USART_DR)。 0：没有检测到空闲总线； 1：检测到空闲总线。 注意：IDLE 位不会再次被置高直到 RXNE 位被置起(即又检测到一次空闲总线)
位 3	**ORE**：过载错误(Overrun error) 当 RXNE 仍然是'1'的时候，当前被接收在移位寄存器中的数据，需要传送至 RDR 寄存器时，硬件将该位置位。如果 USART_CR1 中的 RXNEIE 为'1'的话，则产生中断。由软件序列将其清零(先读 USART_SR，然后读 USART_CR)。 0：没有过载错误； 1：检测到过载错误。 注意：该位被置位时，RDR 寄存器中的值不会丢失，但是移位寄存器中的数据会被覆盖。如果设置了 EIE 位，在多缓冲器通信模式下，ORE 标志置位会产生中断的。
位 2	**NE**: 噪声错误标志(Noise error flag) 在接收到的帧检测到噪音时，由硬件对该位置位。由软件序列对其清玲(先读 USART_SR，再读 USART_DR)。 0：没有检测到噪声； 1：检测到噪声。 注意：该位不会产生中断，因为它和 RXNE 一起出现，硬件会在设置 RXNE 标志时产生中断。在多缓冲区通信模式下，如果设置了 EIE 位，则设置 NE 标志时会产生中断。
位 1	**FE**: 帧错误(Framing error) 当检测到同步错位，过多的噪声或者检测到断开符，该位被硬件置位。由软件序列将其清零(先读 USART_SR，再读 USART_DR)。 0：没有检测到帧错误； 1：检测到帧错误或者 break 符。 注意：该位不会产生中断，因为它和 RXNE 一起出现，硬件会在设置 RXNE 标志时产生中断。如果当前传输的数据既产生了帧错误，又产生了过载错误，硬件还是会继续该数据的传输，并且只设置 ORE 标志位。 在多缓冲区通信模式下，如果设置了 EIE 位，则设置 FE 标志时会产生中断。
位 0	**PE**: 校验错误(Parity error) 在接收模式下，如果出现奇偶校验错误，硬件对该位置位。由软件序列对其清零(依次读 USART_SR 和 USART_DR)。在清除 PE 位前，软件必须等待 RXNE 标志位被置'1'。如果 USART_CR1 中的 PEIE 为'1'，则产生中断。 0：没有奇偶校验错误； 1：奇偶校验错误。

21.6.2 数据寄存器(USART_DR)

地址偏移：0x04

复位值：不确定

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留							DR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:9	保留位，硬件强制为 0
位 8:0	**DR[8:0]**：数据值(Data value) 包含了发送或接收的数据。由于它是由两个寄存器组成的，一个给发送用(TDR)，一个给接收 用(RDR)，该寄存器兼具读和写的功能。TDR 寄存器提供了内部总线和输出移位寄存器之间的并行接口(参见图 173)。RDR 寄存器提供了输入移位寄存器和内部总线之间的并行接口。 当使能校验位(USART_CR1 中 PCE 位被置位)进行发送时，写到 MSB 的值(根据数据的长度不同，MSB 是第 7 位或者第 8 位)会被后来的校验位该取代。 当使能校验位进行接收时，读到的 MSB 位是接收到的校验位。

21.6.3 波特比率寄存器(USART_BRR)

注意： *如果 TE 或 RE 被分别禁止，波特计数器停止计数*

地址偏移：0x08

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV_Mantissa[11:0]												DIV_Fraction[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16	保留位，硬件强制为 0
位 15:4	**DIV_Mantissa[11:0]**：USARTDIV 的整数部分 这 12 位定义了 USART 分频器除法因子(USARTDIV)的整数部分。
位 3:0	**DIV_Fraction[3:0]**：USARTDIV 的小数部分 这 4 位定义了 USART 分频器除法因子(USARTDIV)的小数部分。

21.6.4 控制寄存器 1(USART_CR1)

地址偏移：0x0C

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:14	保留位，硬件强制为 0。
位 13	**UE**：USART 使能(USART enable) 当该位被清零，在当前字节传输完成后 USART 的分频器和输出停止工作，以减少功耗。该位由软件设置和清零。 0：USART 分频器和输出被禁止； 1：USART 模块使能。
位 12	**M**：字长(Word length) 该位定义了数据字的长度，由软件对其设置和清零 0：一个起始位，8 个数据位，n 个停止位； 1：一个起始位，9 个数据位，n 个停止位。 注意：在数据传输过程中(发送或者接收时)，不能修改这个位。
位 11	**WAKE**：唤醒的方法(Wakeup method) 这位决定了把 USART 唤醒的方法，由软件对该位设置和清零。 0：被空闲总线唤醒； 1：被地址标记唤醒。

位 10	**PCE**：检验控制使能(Parity control enable) 用该位选择是否进行硬件校验控制(对于发送来说就是校验位的产生；对于接收来说就是校验位的检测)。当使能了该位，在发送数据的最高位(如果 M=1，最高位就是第 9 位；如果 M=0，最高位就是第 8 位)插入校验位；对接收到的数据检查其校验位。软件对它置'1'或清'0'。一旦设置了该位，当前字节传输完成后，校验控制才生效。 0：禁止校验控制； 1：使能校验控制。
位 9	**PS**：校验选择(Parity selection) 当校验控制使能后，该位用来选择是采用偶校验还是奇校验。软件对它置'1'或清'0'。当前字节传输完成后，该选择生效。 0：偶校验； 1：奇校验。
位 8	**PEIE**：PE 中断使能(PE interrupt enable) 该位由软件设置或清除。 0：禁止产生中断； 1：当 USART_SR 中的 PE 为'1'时，产生 USART 中断。
位 7	**TXEIE**：发送缓冲区空中断使能(TXE interrupt enable) 该位由软件设置或清除。 0：禁止产生中断； 1：当 USART_SR 中的 TXE 为'1'时，产生 USART 中断。

位 6	**TCIE**：发送完成中断使能(Transmission complete interrupt enable) 该位由软件设置或清除。 0：禁止产生中断； 1：当 USART_SR 中的 TC 为'1'时，产生 USART 中断。
位 5	**RXNEIE**：接收缓冲区非空中断使能(RXNE interrupt enable) 该位由软件设置或清除。 0：禁止产生中断； 1：当 USART_SR 中的 ORE 或者 RXNE 为'1'时，产生 USART 中断。
位 4	**IDLEIE**：IDLE 中断使能(IDLE interrupt enable) 该位由软件设置或清除。 0：禁止产生中断； 1：当 USART_SR 中的 IDLE 为'1'时，产生 USART 中断。
位 3	**TE**：发送使能(Transmitter enable) 该位使能发送器。该位由软件设置或清除。 0：禁止发送； 1：使能发送。 注意：1．在数据传输过程中，除了在智能卡模式下，如果 TE 位上有个 0 脉冲(即设置为'0'之后再设置为'1')，会在当前数据字传输完成后，发送一个"前导符"(空闲总线)。 2．当 TE 被设置后，在真正发送开始之前，有一个比特时间的延迟。
位 2	**RE**：接收使能(Receiver enable) 该位由软件设置或清除。 0：禁止接收； 1：使能接收，并开始搜寻 RX 引脚上的起始位。
位 1	**RWU**：接收唤醒(Receiver wakeup) 该位用来决定是否把 USART 置于静默模式。该位由软件设置或清除。当唤醒序列到来时，硬件也会将其清零。 0：接收器处于正常工作模式； 1：接收器处于静默模式。 注意：1．在把 USART 置于静默模式(设置 RWU 位)之前，USART 要已经先接收了一个数据字节。否则在静默模式下，不能被空闲总线检测唤醒。 2．当配置成地址标记检测唤醒(WAKE 位=1)，在 RXNE 位被置位时，不能用软件修改 RWU 位。
位 0	**SBK**：发送断开帧(Send break) 使用该位来发送断开字符。该位可以由软件设置或清除。操作过程应该是软件设置位它，然后在断开帧的停止位时，由硬件将该位复位。 0：没有发送断开字符； 1：将要发送断开字符。

21.6.5 控制寄存器 2(USART_CR2)

地址偏移：0x10

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	保留	LBDIE	LBDL	保留	ADD[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw		rw	rw	rw	rw

位 31:15	保留位，硬件强制为 0。
位 14	**LINEN**：LIN 模式使能(LIN mode enable) 该位由软件设置或清除。 0：禁止 LIN 模式； 1：使能 LIN 模式。 在 LIN 模式下，可以用 USART_CR1 寄存器中的 SBK 位发送 LIN 同步断开符(低 13 位)，以及检测 LIN 同步断开符。
位 13:12	**STOP**：停止位(STOP bits) 这 2 位用来设置停止位的位数 00：1 个停止位； 01：0.5 个停止位； 10：2 个停止位； 11：1.5 个停止位；
位 11	**CLKEN**：时钟使能(Clock enable) 该位用来使能 CK 引脚 0：禁止 CK 引脚； 1：使能 CK 引脚。
位 10	**CPOL**：时钟极性(Clock polarity) 在同步模式下，可以用该位选择 CK 引脚上时钟输出的极性。和 CPHA 位一起配合来产生需要的时钟/数据的采样关系 0：总线空闲时 CK 引脚上保持低电平； 1：总线空闲时 CK 引脚上保持高电平。
位 9	**CPHA**：时钟相位(Clock phase) 在同步模式下，可以用该位选择 CK 引脚上时钟输出的相位。和 CPOL 位一起配合来产生需要的时钟/数据的采样关系(参见图 184 和图 185)。 0：在时钟的第一个边沿进行数据捕获； 1：在时钟的第二个边沿进行数据捕获。

位 8	**LBCL**：最后一位时钟脉冲(Last bit clock pulse) 在同步模式下，使用该位来控制是否在 CK 引脚上输出最后发送的那个数据字节(MSB)对应的时钟脉冲 0：最后一位数据的时钟脉冲不从 CK 输出； 1：最后一位数据的时钟脉冲会从 CK 输出。 注意：1．最后一个数据位就是第 8 或者第 9 个发送的位(根据 USART_CR1 寄存器中的 M 位所定义的 8 或者 9 位数据帧格式)。
位 7	保留位，硬件强制为 0
位 6	**LBDIE**：LIN 断开符检测中断使能(LIN break detection interrupt enable) 断开符中断屏蔽(使用断开分隔符来检测断开符) 0：禁止中断； 1：只要 USART_SR 寄存器中的 LBD 为'1'就产生中断。
位 5	**LBDL**：LIN 断开符检测长度(LIN break detection length) 该位用来选择是 11 位还是 10 位的断开符检测 0：10 位的断开符检测； 1：11 位的断开符检测。
位 4	保留位，硬件强制为 0
位 3:0	**ADD[3:0]**：本设备的 USART 节点地址 该位域给出本设备 USART 节点的地址。 这是在多处理器通信下的静默模式中使用的，使用地址标记来唤醒某个 USART 设备。

注意：　在使能发送后不能改写这三个位(CPOL、CPHA、LBCL)。

21.6.6 控制寄存器 3(USART_CR3)

地址偏移：0x14

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:11	保留位，硬件强制为 0
位 10	**CTSIE**：CTS 中断使能(CTS interrupt enable) 0：禁止中断； 1：USART_SR 寄存器中的 CTS 为'1'时产生中断。
位 9	**CTSE**：CTS 使能(CTS enable) 0：禁止 CTS 硬件流控制； 1：CTS 模式使能，只有 CTS 输入信号有效(拉成低电平)时才能发送数据。如果在数据传输的过程中，CTS 信号变成无效，那么发完这个数据后，传输就停止下来。如果当 CTS 为无效时，往数据寄存器里写数据，则要等到 CTS 有效时才会发送这个数据。

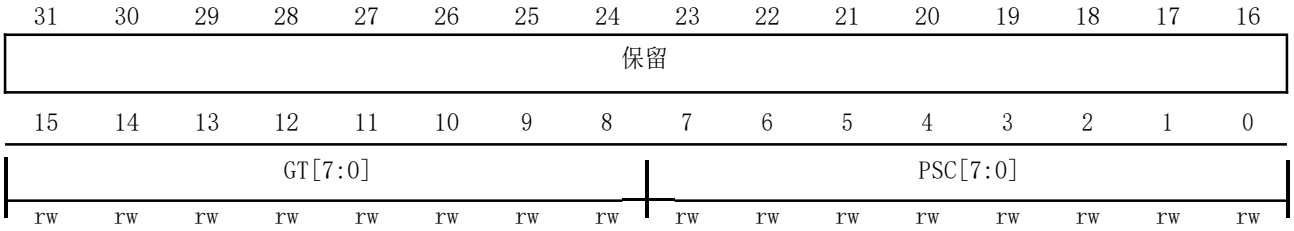
位 8	**RTSE**：RTS 使能(RTS enable) 0：禁止 RTS 硬件流控制； 1：RTS 中断使能，只有接收缓冲区内有空余的空间时才请求下一个数据。当前数据发送完成后，发送操作就需要暂停下来。如果可以接收数据了，将 RTS 输出置为有效(拉至低电平)。注：

位 7	**DMAT**：DMA 使能发送(DMA enable transmitter) 该位由软件设置或清除。 0：禁止发送时的 DMA 模式。 1：使能发送时的 DMA 模式；
位 6	**DMAR**: DMA 使能接收(DMA enable receiver) 该位由软件设置或清除。 0：禁止接收时的 DMA 模式。 1：使能接收时的 DMA 模式；
位 5	**SCEN**: 智能卡模式使能(Smartcard mode enable) 该位用来使能智能卡模式 0：禁止智能卡模式； 1：使能智能卡模式。
位 4	**NACK**：智能卡 NACK 使能(Smartcard NACK enable) 0：校验错误出现时，不发送 NACK； 1：校验错误出现时，发送 NACK。
位 3	**HDSEL**：半双工选择(Half-duplex selection) 选择单线半双工模式 0：不选择半双工模式； 1：选择半双工模式。
位 2	**IRLP**：红外低功耗(IrDA low-power) 该位用来选择普通模式还是低功耗红外模式 0：通常模式； 1：低功耗模式。
位 1	**IREN**：红外模式使能(IrDA mode enable) 该位由软件设置或清除。 0：不使能红外模式； 1：使能红外模式。
位 0	**EIE**：错误中断使能(Error interrupt enable) 在多缓冲区通信模式下，当有帧错误、过载或者噪声错误时(USART_SR 中的 FE=1，或者 ORE=1，或者 NE=1)产生中断。 0：禁止中断； 1：只要 USART_CR3 中的 DMAR=1，并且 USART_SR 中的 FE=1，或者 ORE=1，或者 NE=1，则产生中断

21.6.7 保护时间和预分频寄存器(USART_GTPR)

地址偏移：0x18

复位值：0x0000



位 31:16	保留位，硬件强制为 0
位 15:8	**GT[7:0]**：保护时间值(Guard time value) 该位域规定了以波特时钟为单位的保护时间。在智能卡模式下，需要这个功能。当保护时间过去后，才会设置发送完成标志。
位 7:0	**PSC[7:0]**：预分频器值(Prescaler value) **- 在红外(IrDA)低功耗模式下：** PSC[7:0]=红外低功耗波特率 对系统时钟分频以获得低功耗模式下的频率： 源时钟被寄存器中的值(仅有 8 位有效)分频 00000000：保留– 不要写入该值； 00000001：对源时钟 1 分频； 00000010：对源时钟 2 分频； …… **- 在红外(IrDA)的正常模式下**：PSC 只能设置为 00000001 **- 在智能卡模式下：** PSC[4:0]：预分频值 对系统时钟进行分频，给智能卡提供时钟。 寄存器中给出的值(低 5 位有效)乘以 2 后，作为对源时钟的分频因子 00000：保留– 不要写入该值； 00001：对源时钟进行 2 分频； 00010：对源时钟进行 4 分频； 00011：对源时钟进行 6 分频； …… 注意：1．位[7:5]在智能卡模式下没有意义。

21.6.8 USART 寄存器地址映象

表 88 USART 寄存器列表及其复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	USART_SR	保留																						CTS	LBD	TXEIE	TC	RXNE	IDLE	ORE	NE	FE	PE
	复位值																							0	0	1	1	0	0	0	0	0	0
004h	USART_DR	保留																								DR[8:0]							
	复位值																									0	0	0	0	0	0	0	0
008h	USART_BRR	保留																DIV_Mantissa[15:4]												DIV_Fraction[3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	USART_CR1	保留																		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
	复位值																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	USART_CR2	保留																	LIEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	保留	LBDIE	LBDL	保留	ADD[3:0]			
	复位值																		0	0	0	0	0	0	0		0	0		0	0	0	0
014h	USART_CR3	保留																					CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	复位值																						0	0	0	0	0	0	0	0	0	0	0
018h	USART_GTPR	保留																		GT[7:0]								PSC[7:0]					
	复位值																			0	0	0	0	0	0	0	0	0	0	0	0	0	0

关于寄存器的起始地址，参见表 1。

22 算术运算协处理器 (CO_ALU)

22.1 简介

本协处理器的目的是在原有 ARM Cortex-M3 运算能力的基础上，提高芯片的运算能力，使芯片能使用更多的运算场景。本协处理器在 ARM Cortex-M3 指令的基础上扩充了定点运算和增加了浮点运算。具体的有：

定点运算：

- 64 位、32 位开方运算；
- 64/32 位除法运算；
- 饱和运算：
  - 乘累加饱和运算；
  - 32 位饱和加减运算；
  - SIMD 饱和加减运算
  - 数据饱和化；
- SIMD 加减运算；
- 加减取半运算；
- 扩展加减运算；
- 乘累加运算、SIMD 乘累加运算。

浮点运算：

- 浮点加法运算；
- 浮点减法运算；
- 浮点乘法运算；
- 浮点除法运算；
- 浮点开方运算；
- 浮点乘累加运算；
- 浮点数定点数相互转换运算(包括 32 位定点数和浮点数相互转换、64 位定点数和浮点数相互转换)

22.2 CO_ALU 寄存器

寄存器基址 0x40030000

所有寄存器只支持字访问。寄存器列表如下：

表 34-1 寄存器列表

寄存器	描述
COALU_CR	控制寄存器
COALU_LINK0	通用数据寄存器映射控制寄存器 0
COALU_LINK1	通用数据寄存器映射控制寄存器 1
COALU_SR	状态寄存器
COALU_FR	标志寄存器
COALU_SIMDFR	SIMD 指令标志寄存器
COALU_R0	通用寄存器 0
COALU_R1	通用寄存器 1
COALU_R2	通用寄存器 2
COALU_R3	通用寄存器 3

COALU_R4	通用寄存器 4
COALU_R5	通用寄存器 5
COALU_R6	通用寄存器 6
COALU_R7	通用寄存器 7
COALU_R8	通用寄存器 8
COALU_R9	通用寄存器 9
COALU_R10	通用寄存器 10
COALU_R11	通用寄存器 11

注：CO_ALU 中不同运算需要的操作数寄存器和结果寄存器数量各不相同，最多可能需要 5 个操作数寄存器、3 个结果寄存器。操作数寄存器分别标识为 OP0~OP5，结果寄存器分别标识为 RSLT0~RSLT3。

可以通过配置 COALU_LINK0 或 COALU_LINK1 寄存器把每个操作数寄存器和结果寄存器映射到任意一个通用寄存器。

在一次运算中选择 COALU_LINK0 或者是 COALU_LINK1 中的配置完成映射由 COALU_CR.LK 位选择。如果 COALU_CR.LK 为 0，选择 COALU_LINK0 中的配置完成映射，否则由 COALU_LINK1 中的配置完成映射。

22.2.1 控制寄存器 (COALU_CR)

本寄存器控制 ALU 协处理器的具体操作。当 COALU_SR.BSY 为 1 时访问本寄存器会 stall AHB 总线。

偏移地址：0x00

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留			DIVOFIE	DZIE	IOIE	QFIE	OFIE	保留		RMODE [1:0]		保留		START	DFS
			R/W	R/W	R/W	R/W	R/W			R/W				W	R/W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LK	FAMW	保留				ISA_CODE[9:0]									
R/W	R/W					R/W									

位 31:29	保留
位 28	**DIVOFIE**：除法溢出中断使能 0：无作用 1：使能除法溢出中断
位 27	**DZIE:** 除数为 0 中断使能 0：无作用 1：使能除 0 中断
位 26	**IOIE:** 非法操作中断使能 0：无作用 1：使能非法操作中断
位 25	**QFIE:** 饱和异常中断使能 0：无作用 1：使能饱和异常中断

位 24	**OFIE:** 溢出异常中断使能 0：无作用 1：使能溢出溢出中断
位 23:22	保留
位 21:20	**RMODE:** 用于浮点运算进位控制 00：舍入到最近（RN）模式（如果 GRS>100,在 23 位尾数加 1；如果 GRS<100,丢弃；如果 GSR=100，则如果 23 位尾数的最低位为 1，加 1，如果 23 位尾数的最低位为 0，丢弃） 01：舍入到正无穷（RP）模式（如果结果为正并且 GRS 不等于 0，尾数加 1） 10：舍入到负无穷（RM）模式（如果结果为负并且 GRS 不等于 0，尾数加 1）
位 19:18	保留
位 17	**START：** 软件写 1 开始运算，在 DFS 为 1 时有用。 软件读时始终返回值 0
位 16	**DFS：**Disable Fast Start 默认情况下，软件往 OP0 写数后立即开始运算，当 DFS 为 1 后，需要软件往 START 写 1 后才开始运算。
位 15	**LK：**操作数连接寄存器选择信号。 0：当前指令选择 COALU_LINK0 中的配置映射通用寄存器 1：当前指令选择 COALU_LINK1 中的配置映射通用寄存器
位 14	**FAMW：**浮点数减法、浮点乘法运算、浮点定点数转换 wait 控制。 1：浮点数加减法、浮点乘法、浮点定点数转换运算在 2 个时钟周期内完成。 0：浮点数加减法、浮点乘法、浮点定点数转换运算在 1 个时钟周期内完成。 当 HCLK<60MHz 时可以配置 FAMW 为 0，否则需要配置 FAMW 为 1。
位 13:10	保留
位 9:0	**ISA_CODE[9:0]：**instruction code 指明协处理完成何种运算操作 详细的运用请参考下表

操作定义如下：

	ISA_CODE [9:0]	指令	描述
定点开方	0001_000000	SQRT	32 为无符号定点数开方运算 RSLT0=sqrt(OP0) 本指令不更新状态寄存器
	0001_000001	LSQRT	64 位无符号定点数开方运算 RSLT0=sqrt(OP1, OP0) 本指令不更新状态寄存器
64 位定点除法	0010_000000	SLDIV	带符号数除法，64 位被除数，32 位除数 {RSLT1,RSLT0}={OP2,OP1}/OP0; RSLT2={OP2,OP1}%OP0 本指令可能会更新 SR.DZINT 和 SR.DIVOFINT
	0010_000001	ULDIV	无符号数除法，64 位被除数，32 位除数 {RSLT1,RSLT0}={OP3,OP2}/OP0; RSLT2={OP3,OP2}%OP0 本指令可能会更新 SR.DZINT

SIMD 加减	0011_000000	SADD16	把操作数分解成两个 16 位带符号数，然后相加： RSLT0[31:16]=OP1[31:16]+OP0[31:16], RSLT0[15:0]=OP1[15:0]+OP0[15:0] 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
	0011_000001	SADD8	把操作数分解成 4 个 8 位带符号数，然后相加： RSLT0[31:24]=OP1[31:24]+OP0[31:24, RSLT0[23:16]=OP1[23:16]+OP0[23:16], RSLT0[15:8]=OP1[15:8]+OP0[15:8], RSLT0[7:0]=OP1[7:0]+OP0[7:0] 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
	0011_000010	SSUB16	把操作数分解成两个 16 位带符号数，然后相减： RSLT0[31:16]=OP1[31:16]-OP0[31:16], RSLT0[15:0]=OP1[15:0]-OP0[15:0] 除 Q0~Q3 的其他位
	0011_000011	SSUB8	把操作数分解成 4 个 8 位带符号数，然后相减： RSLT0[31:24]=OP1[31:24]-OP0[31:24, RSLT0[23:16]=OP1[23:16]-OP0[23:16], RSLT0[15:8]=OP1[15:8]-OP0[15:8], RSLT0[7:0]=OP1[7:0]-OP0[7:0] 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
	0011_000100	SASX	把操作数分解成两个 16 位带符号数，然后错位加减： RSLT0[31:16]=OP1[31:16]+OP0[15:0], RSLT0[15:0]=OP1[15:0]-OP0[31:16] 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
	0011_000101	SSAX	把操作数分解成两个 16 位带符号数，然后错位减加： RSLT0[31:16]=OP1[31:16]-OP0[15:0], RSLT0[15:0]=OP1[15:0]+OP0[31:16]; 本指令可能会更新 SIMDFR 状态寄存器 Q0~Q3 的其他位
	0011_000110	UADD16	把操作数分解成两个 16 位无符号数，然后相加： RSLT0[31:16]=OP1[31:16]+OP0[31:16], RSLT0[15:0]=OP1[15:0]+OP0[15:0] 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
	0011_000111	UADD8	把操作数分解成 4 个 8 位无符号数，然后相加： RSLT0[31:24]=OP1[31:24]+OP0[31:24, RSLT0[23:16]=OP1[23:16]+OP0[23:16], RSLT0[15:8]=OP1[15:8]+OP0[15:8], RSLT0[7:0]=OP1[7:0]+OP0[7:0] 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
	0011_001000	USUB16	把操作数分解成两个 16 位无符号数，然后相减： RSLT0[31:16]=OP1[31:16]-OP0[31:16], RSLT0[15:0]=OP1[15:0]-OP0[15:0] 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
	0011_001001	USUB8	把操作数分解成 4 个 8 位无符号数，然后相减： RSLT0[31:24]=OP1[31:24]-OP0[31:24, RSLT0[23:16]=OP1[23:16]-OP0[23:16], RSLT0[15:8]=OP1[15:8]-OP0[15:8], RSLT0[7:0]=OP1[7:0]-OP0[7:0] 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位

	0011_001010	UASX	把操作数分解成两个 16 位无符号数，然后错位加减： RSLT0[31:16]=OP1[31:16]+OP0[15:0], RSLT0[15:0]=OP1[15:0]-OP0[31:16] 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
	0011_001011	USAX	把操作数分解成两个 16 位无符号数，然后错位减加： RSLT0[31:16]=OP1[31:16]-OP0[15:0], RSLT0[15:0]=OP1[15:0]+OP0[31:16]; 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
SIMD 饱和加减运算	0100_000000	QADD16	把操作数分解成两个 16 位带符号数饱和加： RSLT0[31:16]=OP1[31:16]+OP0[31:16], RSLT0[15:0]=OP1[15:0]+OP0[15:0] 本指令可能会更新 SIMDFR 状态寄存器
	0100_000001	QADD8	把操作数分解成 4 个 8 位带符号数饱和加： RSLT0[31:24]=OP1[31:24]+OP0[31:24, RSLT0[23:16]=OP1[23:16]+OP0[23:16], RSLT0[15:8]=OP1[15:8]+OP0[15:8], RSLT0[7:0]=OP1[7:0]+OP0[7:0] 本指令可能会更新 SIMDFR 状态寄存器
	0100_000010	QSUB16	把操作数分解成两个 16 位带符号数饱和减： RSLT0[31:16]=OP1[31:16]-OP0[31:16], RSLT0[15:0]=OP1[15:0]-OP0[15:0] 本指令可能会更新 SIMDFR 状态寄存器
	0100_000011	QSUB8	把操作数分解成 4 个 8 位带符号数饱和减： RSLT0[31:24]=OP1[31:24]-OP0[31:24, RSLT0[23:16]=OP1[23:16]-OP0[23:16], RSLT0[15:8]=OP1[15:8]-OP0[15:8], RSLT0[7:0]=OP1[7:0]-OP0[7:0] 本指令可能会更新 SIMDFR 状态寄存器
	0100_000100	QASX	把操作数分解成两个 16 位带符号数，错位饱和加减： RSLT0[31:16]=OP1[31:16]+OP0[15:0], RSLT0[15:0]=OP1[15:0]-OP0[31:16] 本指令可能会更新 SIMDFR 状态寄存器
	0100_000101	QSAX	把操作数分解成两个 16 位带符号数，错位饱和减加： RSLT0[31:16]=OP1[31:16]-OP0[15:0], RSLT0[15:0]=OP1[15:0]+OP0[31:16]; 本指令可能会更新 SIMDFR 状态寄存器
	0100_000110	UQADD16	把操作数分解成两个 16 位无符号数饱和加： RSLT0[31:16]=OP1[31:16]+OP0[31:16], RSLT0[15:0]=OP1[15:0]+OP0[15:0] 本指令可能会更新 SIMDFR 状态寄存器
	0100_000111	UQADD8	把操作数分解成 4 个 8 位无符号数饱和加： RSLT0[31:24]=OP1[31:24]+OP0[31:24, RSLT0[23:16]=OP1[23:16]+OP0[23:16], RSLT0[15:8]=OP1[15:8]+OP0[15:8], RSLT0[7:0]=OP1[7:0]+OP0[7:0] 本指令可能会更新 SIMDFR 状态寄存器

	0100_001000	UQSUB16	把操作数分解成两个 16 位无符号数饱和减： RSLT0[31:16]=OP1[31:16]-OP0[31:16], RSLT0[15:0]=OP1[15:0]-OP0[15:0]
	0100_001001	UQSUB8	把操作数分解成 4 个 8 位无符号数饱和减： RSLT0[31:24]=OP1[31:24]-OP0[31:24, RSLT0[23:16]=OP1[23:16]-OP0[23:16], RSLT0[15:8]=OP1[15:8]-OP0[15:8], RSLT0[7:0]=OP1[7:0]-OP0[7:0] 本指令可能会更新 SIMDFR 状态寄存器
	0100_001010	UQASX	把操作数分解成两个 16 位无符号数，错位饱和加减： RSLT0[31:16]=OP1[31:16]+OP0[15:0], RSLT0[15:0]=OP1[15:0]-OP0[31:16] 本指令可能会更新 SIMDFR 状态寄存器
	0100_001011	UQSAX	把操作数分解成两个 16 位无符号数，错位饱和减加： RSLT0[31:16]=OP1[31:16]-OP0[15:0], RSLT0[15:0]=OP1[15:0]+OP0[31:16]; 本指令可能会更新 SIMDFR 状态寄存器
32 位加减饱和运算	0101_000001	QADD	32 位带符号数饱和加 RSLT0=OP1[31:0]+OP0[31:0] 本指令可能会更新 FR 状态寄存器的 Q、V、N 位和 SR 状态寄存器的 OFINT、QFINT 位
	0101_000010	QSUB	32 位带符号数饱和减 RSLT0=OP1[31:0]-OP0[31:0] 本指令可能会更新 FR 状态寄存器的 Q、V、N 位和 SR 状态寄存器的 OFINT、QFINT 位
	0101_000011	QDADD	32 位被加数乘 2 并饱和化后和加数带符号饱和加 RSLT0=OP1[31:0]*2+OP0[31:0] 本指令可能会更新 FR 状态寄存器的 Q、V、N 位和 SR 状态寄存器的 OFINT、QFINT 位
	0101_000100	QDSUB	32 位减数乘 2 并饱和化后和被减数做带符号饱和减法 RSLT0=OP1[31:0]-OP0[31:0]*2 本指令可能会更新 FR 状态寄存器的 Q、V、N 位和 SR 状态寄存器的 OFINT、QFINT 位
数据饱和变换	0110_000001	SSAT16	把操作数分解成两个 16 位带符号数，然后饱和到设定位： RSLT0[31:16]=OP0[31:16]饱和到 OP1 设定位 RSLT0[15:0]=OP0[15:0]饱和到 OP1 设定位 本指令不更新状态寄存器
	0110_000010	USAT16	把操作数分解成两个 16 位无符号数，然后饱和到设定位： RSLT0[31:16]=OP0[31:16]饱和到 OP1 设定位 RSLT0[15:0]=OP0：w[15:0]饱和到 OP1 设定位 本指令不更新状态寄存器
加减取半运算	0111_000000	SHADD16	把操作数分解成两个 16 位带符号数相加，然后右移 1 位： RSLT0[31:16]={OP1[31:16]+OP0[31:16]} >> 1, RSLT0[15:0]={OP1[15:0]+OP0[15:0]} >> 1 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位

	0111_000001	SHADD8	把操作数分解成 4 个 8 位带符号数相加，然后右移 1 位： RSLT0[31:24]={OP1[31:24]+OP0[31:24} >> 1, RSLT0[23:16]={OP1[23:16]+OP0[23:16]} >> 1, RSLT0[15:8]={OP1[15:8]+OP0[15:8]} >>1, RSLT0[7:0]={OP1[7:0]+OP0[7:0]} >> 1 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
	0111_000010	SHSUB16	把操作数分解成两个 16 位带符号数相减，然后右移 1 位： RSLT0[31:16]={OP1[31:16]-OP0[31:16]} >> 1, RSLT0[15:0]={OP1[15:0]-OP0[15:0]} >> 1 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
	0111_000011	SHSUB8	把操作数分解成 4 个 8 位带符号数相减，然后右移 1 位： RSLT0[31:24]={OP1[31:24]-OP0[31:24} >> 1, RSLT0[23:16]={OP1[23:16]-OP0[23:16]} >> 1, RSLT0[15:8]={OP1[15:8]-OP0[15:8]} >> 1, RSLT0[7:0]={OP1[7:0]-OP0[7:0]} >> 1 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
	0111_000100	SHASX	把操作数分解成两个 16 位带符号数错位加减，然后右移 1 位： RSLT0[31:16]={OP1[31:16]+OP0[15:0]} >> 1, RSLT0[15:0]={OP1[15:0]-OP0[31:16]} >> 1 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
	0111_000101	SHSAX	把操作数分解成两个 16 位带符号数错位减加，然后右移 1 位： RSLT0[31:16]={OP1[31:16]-OP0[15:0]} >> 1, RSLT0[15:0]={OP1[15:0]+OP0[31:16]} >> 1 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
	0111_000110	UHADD16	把操作数分解成两个 16 位无符号数相加，然后右移 1 位： RSLT0[31:16]={OP1[31:16]+OP0[31:16]} >> 1, RSLT0[15:0]={OP1[15:0]+OP0[15:0]} >> 1 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
	0111_000111	UHADD8	把操作数分解成 4 个 8 位无符号数相加，然后右移 1 位： RSLT0[31:24]={OP1[31:24]+OP0[31:24} >> 1, RSLT0[23:16]={OP1[23:16]+OP0[23:16]} >> 1, RSLT0[15:8]={OP1[15:8]+OP0[15:8]} >>1, RSLT0[7:0]={OP1[7:0]+OP0[7:0]} >> 1 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
	0111_001000	UHSUB16	把操作数分解成两个 16 位无符号数相减，然后右移 1 位： RSLT0[31:16]={OP1[31:16]-OP0[31:16]} >> 1, RSLT0[15:0]={OP1[15:0]-OP0[15:0]} >> 1 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
	0111_001001	UHSUB8	把操作数分解成 4 个 8 位无符号数相减，然后右移 1 位： RSLT0[31:24]={OP1[31:24]-OP0[31:24} >> 1, RSLT0[23:16]={OP1[23:16]-OP0[23:16]} >> 1, RSLT0[15:8]={OP1[15:8]-OP0[15:8]} >> 1, RSLT0[7:0]={OP1[7:0]-OP0[7:0]} >> 1 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
	0111_001010	UHASX	把操作数分解成两个 16 位无符号数错位加减，然后右移 1 位： RSLT0[31:16]={OP1[31:16]+OP0[15:0]} >> 1, RSLT0[15:0]={OP1[15:0]-OP0[31:16]} >> 1 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位

	0111_001011	UHSAX	把操作数分解成两个 16 位无符号数错位减加，然后右移 1 位： RSLT0[31:16]={OP1[31:16]-OP0[15:0]} >> 1, RSLT0[15:0]={OP1[15:0]+OP0[31:16]} >> 1 本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位
乘累加	1000_000000	SMLALBB	把被乘数和乘数的低 16 位做带符号乘法，然后和一个 64bit 带符号数累加。 {RSLT1, RSLT0}=OP1[15:0]*OP0[15:0]+{OP3,OP2} 可能会更新 FR.V、FR.N 和 SR.OFINT
	1000_000001	SMLALBT	把被乘数的低 16 位和乘数的高 16 位做带符号乘法,然后和一个 64bit 带符号数累加。 {RSLT1, RSLT0}=OP1[15:0]*OP0[31:16]+{OP3,OP2}
	1000_000010	SMLALTB	把被乘数的高 16 位和乘数的低 16 位做带符号乘法,然后和一个 64bit 带符号数累加。 {RSLT1, RSLT0}=OP1[31:16]*OP0[15:0]+{OP3,OP2} 可能会更新 FR.V、FR.N 和 SR.OFINT
	1000_000011	SMLALTT	把被乘数的高 16 位和乘数的高 16 位做带符号乘法,然后和一个 64bit 带符号数累加。 {RSLT1, RSLT0}=OP1[31:16]*OP0[31:16]+{OP3,OP2} 可能会更新 FR.V、FR.N 和 SR.OFINT
	1000_000100	SMLAW	把被乘数和乘数做带符号数乘法，然后累加 RSLT0=OP1*OP0+OP2 可能会更新 FR.V、FR.N 和 SR.OFINT
	1000_000101	SMLALD	把被乘数和乘数的高低 16 位分别做带符号乘法，然后和一个 64bit 带符号数累加。 {RSLT1, RSLT0}=OP1[31:16]*OP0[31:16]+OP1[15:0]*OP0[15:0]+{OP3,OP2} 可能会更新 FR.V、FR.N 和 SR.OFINT
	1000_000110	SMLALDX	把被乘数和乘数的高低 16 位交叉做带符号乘法，然后和一个 64bit 带符号数累加。 {RSLT1, RSLT0}=OP1[31:16]*OP0[15:0]+OP1[15:0]*OP0[31:16]+{OP3,OP2} 可能会更新 FR.V、FR.N 和 SR.OFINT
	1000_000111	SMLSLD	把被乘数和乘数的高低 16 位分别做带符号乘法，然后乘积相减，再和一个 64bit 带符号数累加。 {RSLT1, RSLT0}=OP1[15:0]*OP0[15:0]-OP1[31:16]*OP0[31:16]+{OP3,OP2}
	1000_001000	SMLSLDX	把被乘数和乘数的高低 16 位交叉做带符号乘法，然后乘积相减，再和一个 64bit 带符号数累加。 {RSLT1, RSLT0}=OP1[15:0]*OP0[32:16]-OP1[31:16]*OP0[15:0]+{OP3,OP2}
	1000_001001	SMMLA	把被乘数和乘数做带符号乘法，然后选取高 32 位结果，再和一个 32bit 带符号数相加。 RSLT1={OP1*OP0}[63:32]+OP3 可能会更新 FR.V、FR.N 和 SR.OFINT
	1000_001010	SMMLAR	把被乘数和乘数做带符号乘法，然后选取高 32 位舍入后的结果，再和一个 32bit 带符号数累加。 RSLT1={OP1*OP0+32'h80000000}[63:32]+OP3 可能会更新 FR.V、FR.N 和 SR.OFINT

	1000_001011	SMMLS	把被乘数和乘数做带符号乘法，然后选取高 32 位结果，再减去一个 32bit 带符号数。 RSLT1={OP1*OP0}[63:32]-OP3
	1000_001100	SMMLSR	把被乘数和乘数做带符号乘法，然后选取高 32 位舍入后的结果，再减去一个 32bit 带符号数。 RSLT1={OP1*OP0+32'h80000000}[63:32]-OP3 可能会更新 FR.V、FR.N 和 SR.OFINT
	1000_001101	UMAAL	把被乘数和乘数做 32 位带符号乘法，并与{OP3, OP2}相加得到 64 位结果 {RSLT1,RSLT0}=OP1*OP0+{OP3,OP2} 可能会更新 FR.V、FR.N 和 SR.OFINT
	1000_001110	SMMAAA	把两组 32 位带符号被乘数和乘数相乘，然后和一个 32 位带符号数累加 RSLT0=OP4*OP3+OP1*OP0+OP2 可能会更新 FR.V、FR.N 和 SR.OFINT
饱和乘累加	1001_000000	QSMLALBB	把被乘数和乘数的低 16 位做带符号乘法，然后和一个 64bit 带符号数累加；结果饱和为 64 位 {RSLT1, RSLT0}=OP1[15:0]*OP0[15:0]+{OP3,OP2} 可能会更新 FR.Q、FR.V、FR.N 和 SR.QFINT、SR.OFINT
	1001_000001	QSMLALBT	把被乘数的低 16 位和乘数的高 16 位做带符号乘法,然后和一个 64bit 带符号数累加；结果饱和为 64 位 {RSLT1, RSLT0}=OP1[15:0]*OP0[31:16]+{OP3,OP2} 可能会更新 FR.Q、FR.V、FR.N 和 SR.QFINT、SR.OFINT
	1001_000010	QSMLALTB	把被乘数的高 16 位和乘数的低 16 位做带符号乘法,然后和一个 64bit 带符号数累加；结果饱和为 64 位 {RSLT1, RSLT0}=OP1[31:16]*OP0[15:0]+{OP3,OP2} 可能会更新 FR.Q、FR.V、FR.N 和 SR.QFINT、SR.OFINT
	1001_000011	QSMLALTT	把被乘数的高 16 位和乘数的高 16 位做带符号乘法,然后和一个 64bit 带符号数累加；结果饱和为 64 位 {RSLT1, RSLT0}=OP1[31:16]*OP0[31:16]+{OP3,OP2} 可能会更新 FR.Q、FR.V、FR.N 和 SR.QFINT、SR.OFINT
	1001_000100	QSMLAW	把被乘数和乘数做带符号数乘法，然后累加；结果饱和为 32 位 RSLT0=OP1*OP0+OP2 可能会更新 FR.Q、FR.V、FR.N 和 SR.QFINT、SR.OFINT
	1001_000101	QSMLALD	把被乘数和乘数的高低 16 位分别做带符号乘法，然后和一个 64bit 带符号数累加；结果饱和为 64 位 {RSLT1, RSLT0}=OP1[31:16]*OP0[31:16]+OP1[15:0]*OP0[15:0]+{OP3,OP2} 可能会更新 FR.Q、FR.V、FR.N 和 SR.QFINT、SR.OFINT
	1001_000110	QSMLALDX	把被乘数和乘数的高低 16 位交叉做带符号乘法，然后和一个 64bit 带符号数累加；结果饱和为 64 位 {RSLT1, RSLT0}=OP1[31:16]*OP0[15:0]+OP1[15:0]*OP0[31:16]+{OP3,OP2} 可能会更新 FR.Q、FR.V、FR.N 和 SR.QFINT、SR.OFINT
	1001_000111	QSMLSLD	把被乘数和乘数的高低 16 位分别做带符号乘法,然后乘积相减，再和一个 64bit 带符号数累加；结果饱和为 64 位 {RSLT1, RSLT0}=OP1[15:0]*OP0[15:0]-OP1[31:16]*OP0[31:16]+{OP3,OP2} 可能会更新 FR.Q、FR.V、FR.N 和 SR.QFINT、SR.OFINT

	1001_001000	QSMLSLDX	把被乘数和乘数的高低 16 位交叉做带符号乘法，然后乘积相减，再和一个 64bit 带符号数累加；结果饱和为 64 位 {RSLT1, RSLT0}=OP1[15:0]*OP0[32:16]-OP1[31:16]*OP0[15:0]+{OP3,OP2}
	1001_001001	QSMMLA	把被乘数和乘数做带符号乘法，然后选取高 32 位结果，再和一个 32bit 带符号数相加；结果饱和为 32 位 RSLT1={OP1*OP0}[63:32]+OP3
	1001_001010	QSMMLAR	把被乘数和乘数做带符号乘法，然后选取高 32 位舍入后的结果，再和一个 32bit 带符号数累加；结果饱和为 32 位 RSLT1={OP1*OP0+32'h80000000}[63:32]+OP3
	1001_001011	QSMMLS	把被乘数和乘数做带符号乘法，然后选取高 32 位结果，再减去一个 32bit 带符号数；结果饱和为 32 位 RSLT1={OP1*OP0}[63:32]-OP3 可能会更新 FR.Q、FR.V、FR.N 和 SR.QFINT、SR.OFINT
	1001_001100	QSMMLSR	把被乘数和乘数做带符号乘法，然后选取高 32 位舍入后的结果，再减去一个 32bit 带符号数；结果饱和为 32 位 RSLT1={OP1*OP0+32'h80000000}[63:32]-OP3 可能会更新 FR.Q、FR.V、FR.N 和 SR.QFINT、SR.OFINT
	1001_001101	QUMAAL	把被乘数和乘数做 32 位带符号乘法，并与{OP4, OP3}相加得到 64 位结果 {RSLT1,RSLT0}=OP1*OP0+{OP3,OP2} 可能会更新 FR.Q、FR.V、FR.N 和 SR.QFINT、SR.OFINT
	1001_001110	QSMMAAA	把两组 32 位带符号被乘数和乘数相乘，然后和一个 32 位带符号数累加，乘法结果和累加结果都饱和化为 32 位 RSLT0=OP4*OP2+OP1*OP0+OP2 可能会更新 FR.Q、FR.V、FR.N 和 SR.QFINT、SR.OFINT
浮点运算	1010_000000	VADD	浮点数加法 RSLT0=OP1+OP0 可能会更新 FR.FV、FR.FZ、FR.FN 和 SR.IOINT
	1010_000001	VSUB.F32	浮点数减法 RSLT0=OP1-OP0 可能会更新 FR.FV、FR.FZ、FR.FN 和 SR.IOINT
	1010_000010	VMUL.F32	浮点数乘法 RSLT0=OP2*OP1 可能会更新 FR.FV、FR.FZ、FR.FN 和 SR.IOINT
	1010_000011	VDIV.F32	浮点数除法 RSLT0=OP1/OP0 可能会更新 FR.FV、FR.FZ、FR.FN 和 SR.IOINT、SR.DZINT
	1010_000100	VSQRT.F32	浮点数开方 RSLT0=sqrt（OP0） 可能会更新 FR.FV、FR.FZ 和 SR.IOINT
	1010_000101	VLSQRT.F32	高精度浮点开方（把尾数补零扩展为 48 位后开方，提高指数为奇数的开方精度） RSLT0=sqrt（OP0） 可能会更新 FR.FV、FR.FZ 和 SR.IOINT
	1010_001001	VCVTR.S32.F32	把浮点数转换为 32 位带符号定点数，舍入方式由 CR.RMODE[1:0]决定 RSLT0=OP0 可能会更新 FR.FV、FR.FZ、FR.FN 和 SR.IOINT

	1010_001010	VCVT.F32.S32	把 32 位带符号定点数转换为浮点数 RSLT0=OP0 可能会更新 FR.FZ、FR.FN
	1010_001111	VMLA.F32	浮点数乘累加，对乘法运算的结果进行舍入后再进行加法 RSLT0=OP2+OP1*OP0 可能会更新 FR.FV、FR.FZ、FR.FN 和 SR.IOINT
	1010_010000	VMLS.F32	浮点数乘减，对乘法运算的结果进行舍入后再进行减法 RSLT0=OP2-OP1*OP0 可能会更新 FR.FV、FR.FZ、FR.FN 和 SR.IOINT
	1010_010010	VNMLA.F32	浮点乘累加并取负数 RSLT0=-（OP2+OP1*OP0） 可能会更新 FR.FV、FR.FZ、FR.FN 和 SR.IOINT
	1010_010011	VNMLS.F32	浮点乘累减并取负数 RSLT0=-（OP2-OP1*OP0） 可能会更新 FR.FV、FR.FZ、FR.FN 和 SR.IOINT
	1010_010100	VNMUL	浮点乘法并取负数 RSLT0=-OP1*OP0 可能会更新 FR.FV、FR.FZ、FR.FN 和 SR.IOINT
	1010_010101	VCVTR.S64.F32	把浮点数转换为 64 位带符号定点数，舍入方式由 CR.RMODE[1:0]决定 {RSLT1, RSLT0}=OP0 可能会更新 FR.FV、FR.FZ、FR.FN 和 SR.IOINT
	1010_010110	VCVT.F32.S64	把 64 位带符号定点数转换为浮点数 RSLT0={OP1, OP0} 可能会更新 FR.FZ、FR.FN

22.2.2 操作数连接寄存器 0(COALU_LINK0)

通过配置本寄存器可以把运算结果直接映射到操作数寄存器，作为下一次运算的输入操作数。当 COALU_SR.BSY 为 1 时访问本寄存器会 stall AHB 总线。

偏移地址：0x04

复位值：0x7654_3210

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16
RSLT2_LK0[3:0]	RSLT1_LK0[3:0]	RSLT0_LK0[3:0]	OP4_LK0[3:0]
R/W	R/W	R/W	R/W

15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
OP3_LK0[3:0]	OP2_LK0[3:0]	OP1_LK0[3:0]	OP0_LK0[3:0]
R/W	R/W	R/W	R/W

位 31:28	**RSLT2_LK0[3:0]:** 0000 - 把 RSLT2 映射到 COALU_R0 0001- 把 RSLT2 映射到 COALU_R1 … 1011- 把 RSLT2 映射到 COALU_R11 Others：禁止使用
位 27:24	**RSLT1_LK0[3:0]** 0000 - 把 RSLT1 映射到 COALU_R0 0001- 把 RSLT1 映射到 COALU_R1 … 1011- 把 RSLT1 映射到 COALU_R11 Others：禁止使用
位 23:20	**RSLT0_LK0[3:0]:** 0000 - 把 RSLT0 映射到 COALU_R0 0001- 把 RSLT0 映射到 COALU_R1 … 1011- 把 RSLT0 映射到 COALU_R11 Others：禁止使用
位 19:16	**OP4_LK0[3:0]:** 0000 - 把 OP4 映射到 COALU_R0 0001- 把 OP4 映射到 COALU_R1 … 1011- 把 OP4 映射到 COALU_R11 Others：禁止使用
位 15:12	**OP3_LK0[3:0]:** 0000 - 把 OP3 映射到 COALU_R0 0001- 把 OP3 映射到 COALU_R1 … 1011- 把 OP3 映射到 COALU_R11 Others：禁止使用
位 11:8	**OP2_LK0[3:0]:** 0000 - 把 OP2 映射到 COALU_R0 0001- 把 OP2 映射到 COALU_R1 … 1011- 把 OP2 映射到 COALU_R11 Others：禁止使用
位 7:4	**OP1_LK0[3:0]:** 0000 - 把 OP1 映射到 COALU_R0 0001- 把 OP1 映射到 COALU_R1 … 1011- 把 OP1 映射到 COALU_R11 Others：禁止使用
位 3:0	**OP0_LK0[3:0]:** 0000 - 把 OP0 映射到 COALU_R0 0001- 把 OP0 映射到 COALU_R1 … 1011- 把 OP0 映射到 COALU_R11 Others：禁止使用

注：可以把一个通用寄存器同时映射到多个 OP 寄存器和 RSLT 寄存器。
比如执行指令：RSLT0=OP1*OP0+OP2，如果把 R0 映射到 OP2，OP1 和 RSLT0，把 R1 映射到 OP0，则执行指令后 R0=R0*R1+R0。

22.2.3 操作数连接寄存器 1(COALU_LINK1)

通过配置本寄存器可以把运算结果直接映射到操作数寄存器，作为下一次运算的输入操作数。当 COALU_SR.BSY 为 1 时访问本寄存器会 stall AHB 总线。

偏移地址：0x08

复位值：0xba98_7650

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16
RSLT2_LK1[3:0]	RSLT1_LK1[3:0]	RSLT0_LK1[3:0]	OP4_LK1[3:0]
R/W	R/W	R/W	R/W

15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
OP3_LK1[3:0]	OP2_LK1[3:0]	OP1_LK1[3:0]	OP0_LK1[3:0]
R/W	R/W	R/W	R/W

位 31:28	**RSLT2_LK1[3:0]:** 0000 - 把 RSLT2 映射到 COALU_R0 0001- 把 RSLT2 映射到 COALU_R1 … 1011- 把 RSLT2 映射到 COALU_R11 Others：禁止使用
位 27:24	**RSLT1_LK1[3:0]** 0000 - 把 RSLT1 映射到 COALU_R0 0001- 把 RSLT1 映射到 COALU_R1 … 1011- 把 RSLT1 映射到 COALU_R11 Others：禁止使用
位 23:20	**RSLT0_LK1[3:0]:** 0000 - 把 RSLT0 映射到 COALU_R0 0001- 把 RSLT0 映射到 COALU_R1 … 1011- 把 RSLT0 映射到 COALU_R11 Others：禁止使用
位 19:16	**OP4_LK1[3:0]:** 0000 - 把 OP4 映射到 COALU_R0 0001- 把 OP4 映射到 COALU_R1 … 1011- 把 OP4 映射到 COALU_R11 Others：禁止使用
位 15:12	**OP3_LK1[3:0]:** 0000 - 把 OP3 映射到 COALU_R0 0001- 把 OP3 映射到 COALU_R1 … 1011- 把 OP3 映射到 COALU_R11 Others：禁止使用
位 11:8	**OP2_LK1[3:0]:** 0000 - 把 OP2 映射到 COALU_R0 0001- 把 OP2 映射到 COALU_R1 … 1011- 把 OP2 映射到 COALU_R11 Others：禁止使用

位 7:4	**OP1_LK1[3:0]:** 0000 - 把 OP1 映射到 COALU_R0 0001- 把 OP1 映射到 COALU_R1 … 1011- 把 OP1 映射到 COALU_R11 Others：禁止使用
位 3:0	**OP0_LK1[3:0]:** 0000 - 把 OP0 映射到 COALU_R0 0001- 把 OP0 映射到 COALU_R1 … 1011- 把 OP0 映射到 COALU_R11 Others：禁止使用

注：可以把一个通用寄存器同时映射到多个 OP 寄存器和 RSLT 寄存器。
比如执行指令：RSLT0=OP1*OP0+OP2，如果把 R0 映射到 OP2，OP1 和 RSLT0，把 R1 映射到 OP0，则执行指令后 R0=R0*R1+R0。

22.2.4 状态寄存器(COALU_SR)

通过配置本寄存器可以把运算结果直接映射到操作数寄存器，作为下一次运算的输入操作数。软件可以轮询 BSY 位判断运算是否结束。当 BSY 为 1 时，写本寄存器的其他位会 stall AHB 总线。

偏移地址：0x0c

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留											DIV_OFINT	DZINT	IOINT	QFINT	OFINT
											R/W	R/W	R/W	R/W	R/W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留															BSY
															R

位 31:21	保留
位 20	DIV_OFINT： 硬件跟根据运算数据置 1 或清零该位 软件往该位写 0 可清零，软件不能置 1 该位 带符号除法中，如果被除数为 0x8000_0000_0000_0000，除数为 0xffff_ffff，这种情况下会置位 DIV_OFINT。这时所得的商为 0x8000_0000_0000_0000，余数为 0。
位 19	**DZINT:** 硬件跟根据运算数据置 1 或清零该位 软件往该位写 0 可清零，软件不能置 1 该位被 0 除累积异常中断标志（浮点运算或普通运算都会更新）
位 18	**IOINT:** 硬件跟根据运算数据置 1 或清零该位 软件往该位写 0 可清零，软件不能置 1 该位非法操作异常中断标志（浮点数操作专用，结果是 NaN 时置位 IOINT）

位 17	**QFINT:** 硬件跟根据运算数据置 1 或清零该位 软件往该位写 0 可清零，软件不能置 1 该位饱和 异常中断标志，标识运算结果被饱和
位 16	**OFINT:** 硬件跟根据运算数据置 1 或清零该位 软件往该位写 0 可清零，软件不能置 1 该位溢出异常中断标志（浮点运算溢出或普通运算溢出）
位 15:1	保留
位 0	**BSY:** 1：表示协处理中运算正在进行 0：表示协处理器当前空闲 BSY 信号由邮件置位和清零，不可被软件改写

如果 COALU_SR 中的 xINT 位被置 1，同时 COALU_CR 中的中断使能位也为 1，则发送 COALU 中断请求。

22.2.5 标志寄存器(COALU_FR)

通过配置本寄存器可以把运算结果直接映射到操作数寄存器，作为下一次运算的输入操作数。当 COALU_SR.BSY 为 1 时访问本寄存器会 stall AHB 总线。

偏移地址：0x10

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留												FV	保留	FZ	FN
												R/W		R/W	R/W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留											Q	V	保留		N
											R/W	R/W			R/W

位 31:20	保留
位 19	**FV:**浮点运算溢出标志 硬件跟根据运算数据置 1 或清零该位 软件往该位写 0 可清零，软件不能置 1 该位
位 18	保留
位 17	**FZ:**浮点运算结果等于 0 标志 硬件跟根据运算数据置 1 或清零该位 软件往该位写 0 可清零，软件不能置 1 该位
位 16	**FN:**浮点运算结果为负标志 硬件跟根据运算数据置 1 或清零该位 软件往该位写 0 可清零，软件不能置 1 该位
位 15:5	保留
位 4	**Q：**饱和标志，标识运算结果被饱和 硬件跟根据运算数据置 1 或清零该位 软件往该位写 0 可清零，软件不能置 1 该位
位 3	**V：**溢出标志，标识运算结果发生溢出 硬件跟根据运算数据置 1 或清零该位 软件往该位写 0 可清零，软件不能置 1 该位

位 2:1	保留
位 0	**N**：负标志，标识运算结果为负数 硬件跟根据运算数据置 1 或清零该位 软件往该位写 0 可清零，软件不能置 1 该位

注意： *1. 不是所有运算都会更新 Q，V，N 标志。*
2. 当一次运算还没有完成时，如果读标志寄存器会 halt 总线直到运算完成。
3. 软件可以把标志位清零，不能置 1。

22.2.6 SIMD 指令标志寄存器(COALU_SIMDFR)

通过配置本寄存器可以把运算结果直接映射到操作数寄存器，作为下一次运算的输入操作数。当 COALU_SR.BSY 为 1 时访问本寄存器会 stall AHB 总线。

偏移地址：0x14

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留												GE3	GE2	GE1	GE0
												R/W	R/W	R/W	R/W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Q3	Q2	Q1	Q0	V3	V2	V1	V0	Z3	Z2	Z1	Z0	N3	N2	N1	N0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

位 31:20	保留
位 19	**GE3:**SIMD 指令运算结果第 3 段大于 0 标志
位 18	**GE2**：SIMD 指令运算结果第 2 大于 0 标志
位 17	**GE1**：SIMD 指令运算结果第 1 大于 0 标志
位 16	**GE0**：SIMD 指令运算结果第 0 大于 0 标志
位 15	**Q3**：SIMD 指令运算结果第 3 段饱和标志
位 14	**Q2**：SIMD 指令运算结果第 2 饱和标志
位 13	**Q1**：SIMD 指令运算结果第 1 饱和标志
位 12	**Q0**：SIMD 指令运算结果第 0 饱和标志
位 11	**V3**：SIMD 指令运算结果第 3 段溢出标志
位 10	**V2**：SIMD 指令运算结果第 2 段溢出标志
位 9	**V1**：SIMD 指令运算结果第 1 段溢出标志
位 8	**V0**：SIMD 指令运算结果第 0 段溢出标志
位 7	**Z3**：SIMD 指令运算结果第 3 段为零标志
位 6	**Z2**：SIMD 指令运算结果第 2 段为零标志
位 5	**Z1**：SIMD 指令运算结果第 1 段为零标志
位 4	**Z0**：SIMD 指令运算结果第 0 段为零标志
位 3	**N3**：SIMD 指令运算结果第 3 段为负标志
位 2	**N2**：SIMD 指令运算结果第 2 段为负标志
位 1	**N1**：SIMD 指令运算结果第 1 段为负标志
位 0	**N0**：SIMD 指令运算结果第 0 段为负标志

注意： *1. 不是所有运算都会更新标志位*
2. 若 SIMD 指令为半字运算，则 x3，x2 两个位表示运算结果的高半字状态，并且 x3 和 x2 位相等。
3. 若 SIMD 指令为字节运算，则 x3，x2，x1，x0 每个 bit 分别结果每个字节的状态。
4. 软件可以把标志位清零，不能置 1。

22.2.7 通用寄存器 0~11 (COALU_R0~ COALU_R11)

当 COALU_SR.BSY 为 1 时访问本寄存器会 stall AHB 总线。

偏移地址：0x20~0x4c

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W															

用户操作寄存器，可以软件写入 32 位 data，也可以硬件更新数据。
用户把待运算的数据写到通用寄存器，然后运算的结果也可以存储到通用寄存器

22.3 设计描述

运行时间及标志位更新表
非浮点运算：

	指令	描述	标志位更新	运行周期数
定点开方	SQRT	32 为定点数开方运算 RSLT0=sqrt(OP0)	-	2~17
	LSQRT	64 位定点数开方运算 RSLT0=sqrt(OP1, OP0)	-	2~33
64 位定点除法	SLDIV	带符号数除法，64 位被除数，32 位除数 {RSLT1,RSLT0}={OP3,OP2}/OP0; RSLT2={OP3,OP2}%OP0	SR.DZINT	2~33
	ULDIV	无符号数除法，64 位被除数，32 位除数 {RSLT1,RSLT0}={OP3,OP2}/OP0; RSLT2={OP3,OP2}%OP0	SR.DZINT、SR.DIVOFINT	2~33
SIMD 加减	ISA_CODE[9:0]=0xc0~0xcb		更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位	1
SIMD 饱和加减运算	ISA_CODE[9:0]=0x100~0x10b		SIMDFR 状态寄存器	1

32 位加减饱和运算	ISA_CODE[9:0]=0x141~0x144		FR.Q、V、N 位和 SR.OFINT、QFINT 位	1
加减取半运算	ISA_CODE[9:0]=0x1c0~0x1cb		更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位	1
乘累加	包含 1 个乘法运算的乘累加指令		FR.V、FR.N 和 SR.OFINT	1
	包含 2 个乘法运算的乘累加指令		FR.V、FR.N 和 SR.OFINT	2
饱和乘累加	包含 1 个乘法运算的饱和乘累加指令		FR.Q、FR.V、FR.N 和 SR.QFINT、SR.OFINT	1
	包含 2 个乘法运算的饱和乘累加指令		FR.Q、FR.V、FR.N 和 SR.QFINT、SR.OFINT	2

浮点运算：

	指令	描述	运行周期数
浮点运算	VADD	浮点数加法 RSLT0=OP1+OP0	1~2
	VSUB.F32	浮点数减法 RSLT0=OP1-OP0	1~2
	VMUL.F32	浮点数乘法 RSLT0=OP2*OP1	1~2
	VDIV.F32	浮点数除法 RSLT0=OP1/OP0	14
	VSQRT.F32	浮点数开方 RSLT0=sqrt（OP0）	15
	VLSQRT.F32	高精度浮点开方（把尾数补零扩展为 48 位后开方，提高指数为奇数的开方精度） RSLT0=sqrt（OP0）	26
	VCVTR.S32.F32	把浮点数转换为 32 位带符号定点数，舍入方式由 CR.RMODE[1:0]决定 RSLT0=OP0	1~2
	VCVT.F32.S32	把 32 位带符号定点数转换为浮点数 RSLT0=OP0	1~2
	VMLA.F32	浮点数乘累加，对乘法运算的结果进行舍入后再进行加法 RSLT0=OP2+OP1*OP0	4
	VMLS.F32	浮点数乘减，对乘法运算的结果进行舍入后再进行减法 RSLT0=OP2-OP1*OP0	4

	VNMLA.F32	浮点乘累加并取负数 RSLT0=-（OP2+OP1*OP0）	4
	VNMLS.F32	浮点乘累减并取负数 RSLT0=-（OP2-OP1*OP0）	4
	VNMUL	浮点乘法并取负数 RSLT0=-OP1*OP0	1~2
	VCVTR.S64.F32	把浮点数转换为 64 位带符号定点数，舍入方式由 CR.RMODE[1:0]决定 {RSLT1, RSLT0}=OP0	1~2
	VCVT.F32.S64	把 64 位带符号定点数转换为浮点数 RSLT0={OP1, OP0}	1~2

23 器件电子签名

包含的芯片识别信息在出厂时编写，用户固件或者外部设备可以读取电子签名，用以自动匹配不同配置的 HK32F10xxx 微控制器。

电子签名存放在闪存存储器模块的系统存储区域，可以通过 JTAG/SWD 或者 CPU 读取。它所

23.1 存储器容量寄存器

23.1.1 闪存容量寄存器

基地址：0x1FFF F7E0

只读，它的内容在出厂时编写

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F_SIZE															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 15:0	**F_SIZE**：闪存存储器容量 以 K 字节为单位指示产品中闪存存储器容量。 例：0x0080 = 128 K 字节

23.1.2 产品唯一身份标识寄存器(96 位)

产品唯一的身份标识非常适合：

- 用来作为序列号(例如 USB 字符序列号或者其他的终端应用)
- 用来作为密码，在编写闪存时，将此唯一标识与软件加解密算法结合使用，提高代码在闪存存储器内的安全性。
- 用来激活带安全机制的自举过程。

96 位的产品唯一身份标识所提供的参考号码对任意一个 HK32 微控制器，在任何情况下都是唯一的。用户在何种情况下，都不能修改这个身份标识。

这个 96 位的产品唯一身份标识，按照用户不同的用法，可以以字节(8 位)为单位读取，也可以以半字(16 位)或者全字(32 位)读取。

基地址：0x1FFF F7E8 地址

偏移：0x00

只读，其值在出厂时编写

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 15:0	**U_ID**[15:0]：唯一身份标志 15:0 位

地址偏移：0x02

只读，其值在出厂时编写

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 15:0	**U_ID[31:16]**：唯一身份标志 31:16 位 这个域的数值也预留作为未来的其它功能。

地址偏移：0x04

只读，其值在出厂时编写

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 15:0	**U_ID[63:32]**：唯一身份标志 63:32 位

地址偏移：0x08

只读，其值在出厂时编写

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 15:0	**U_ID[95:64]**：唯一身份标志 95:64 位

24 调试支持（DBG）

24.1 概况

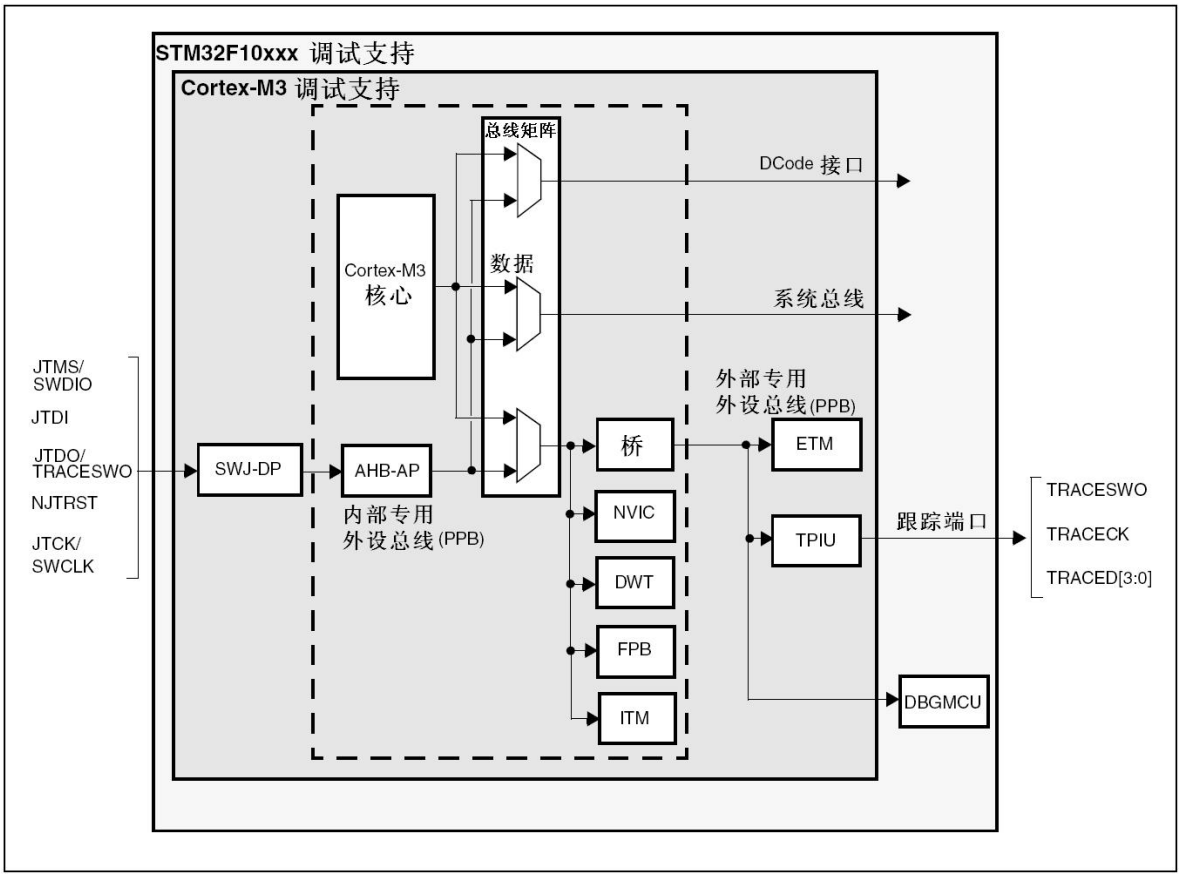
HK32F10xxx 使用 Cortex™-M3 内核，该内核内含硬件调试模块，支持复杂的调试操作。硬件调试模块允许内核在取指(指令断点)或访问数据(数据断点)时停止。内核停止时，内核的内部状态和系统的外部状态都是可以查询的。完成查询后，内核和外设可以被复原，程序将继续执行。

当 HK32F10x 微控制器连接到调试器并开始调试时，调试器将使用内核的硬件调试模块进行调试操作。

支持两种调试接口：

- 串行接口
- JTAG 调试接口

图 200 HK32F10xxx 级别和 Cortex™-M3 级别的调试框图



注意： *Cortex™-M3 内核内含的硬件调试模块是 ARM CoreSight 开发工具集的子集。*

ARM Cortex™-M3 内核提供集成的片上调试功能。它由以下部分组成：

- SWJ-DP：串行/JTAG 调试端口
- AHP-AP：AHB 访问端口
- ITM：执行跟踪单元
- FPB：闪存指令断点
- DWT：数据触发
- TPIU：跟踪单元接口(仅支持 SWV 异步模式（TRACESWO 会被使用））

专用于 HK32F10xxx 的调试特性

- 灵活的调试引脚分配
- MCU 调试盒(支持低电源模式，控制外设时钟等)

注意：更多 ARM Cortex™-M3 内核的调试功能信息，请参考 Cortex™-M3(r1p1 版)技术参考手册(TRM)和 CoreSight 开发工具集(r1p0 版)TRM。

24.2 ARM 参考文献

- Cortex™-M3(r1p1 版)技术参考手册(TRM)
- ARM 调试接口 V5
- ARM CoreSight 开发工具集(r1p0 版)技术参考手册

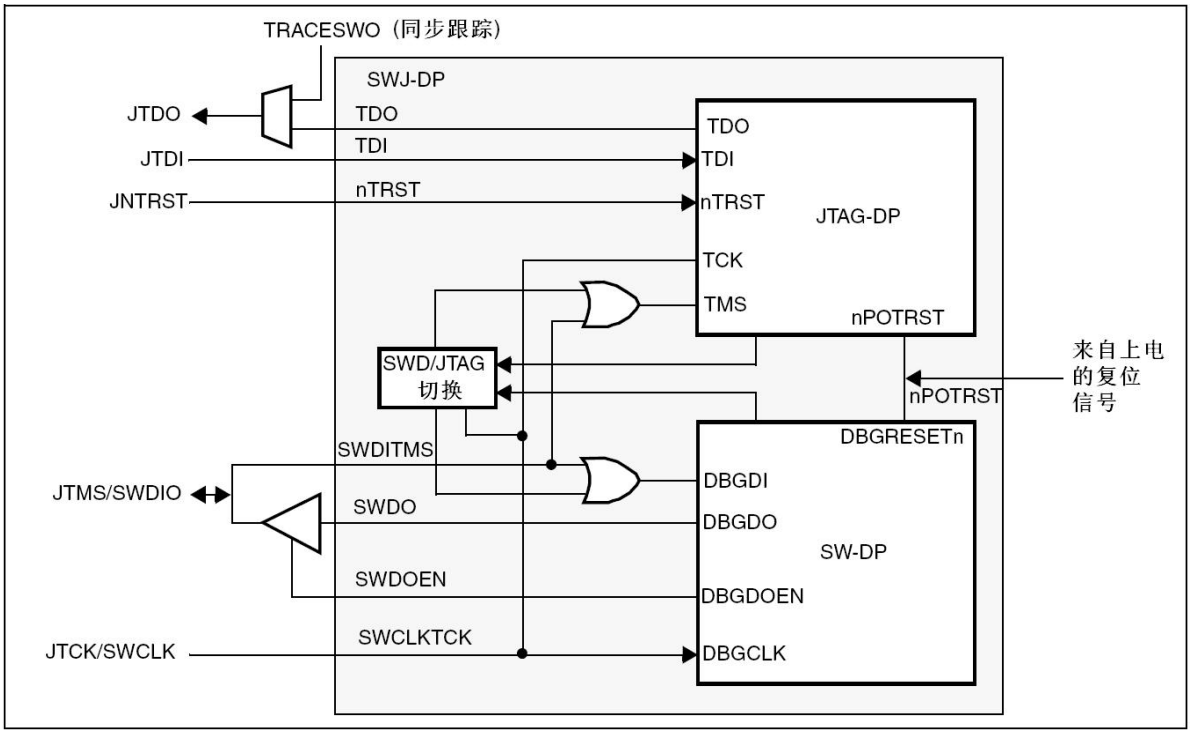
24.3 SWJ 调试端口(serial wire and JTAG)

HK32F10xxx 内核集成了串行/JTAG 调试接口(SWJ-DP)。这是标准的 ARM CoreSight 调试接口，包括 JTAG-DP 接口(5 个引脚)和 SW-DP 接口(2 个引脚)。

- JTAG 调试接口(JTAG-DP)为 AHP-AP 模块提供 5 针标准 JTAG 接口。
- 串行调试接口(SW-DP)为 AHP-AP 模块提供 2 针(时钟＋数据)接口。

在 SWJ-DP 接口中，SW-DP 接口的 2 个引脚和 JTAG 接口的 5 个引脚中的一些是复用的。

图 201 SWJ 调试端口



上面的图显示异步跟踪输出脚(TRACESWO)和 TDO 是复用的，因此异步跟踪功能只能在 SW-DP 调试接口上实现，不能在 JTAG-DP 调试接口上实现。

24.3.1 JTAG-DP 和 SW-DP 切换的机制

JTAG 调试接口是默认的调试接口。

如果调试器想要切换到 SW-DP，必须在 TMS/TCK 上输出一指定的 JTAG 序列(分别映射到 SWDIO 和 SWCLK)，该序列禁止 JTAG-DP，并激活 SW-DP。该方法可以只通过 SWCLK 和 SWDIO 两个引脚来激活 SW-DP 接口。

指定的序列是：

1. 输出超过 50 个 TCK 周期的 TMS(SWDIO)＝1 信号
2. 输出 16 个 TMS(SWDIO)信号 0111100111100111 (MSB)
3. 输出超过 50 个 TCK 周期的 TMS(SWDIO)＝1 信号

24.4 引脚分布和调试端口脚

HK32F10xxx 微控制器的不同封装有不同的有效引脚数。因此，某些与引脚相关的功能可能随封装而不同。

24.4.1 SWJ 调试端口脚

HK32F10xxx 的 5 个普通 I/O 口可用作 SWJ-DP 接口引脚。这些引脚在所有的封装里都存在。

表 93 SWJ 调试端口引脚

SWJ-DP 端口引脚名称	JTAG 调试接口		SW 调试接口		引脚分配
	类型	描述	类型	调试功能	
JTMS/SWDIO	输入	JTAG 模式选择	输入/输出	串行数据输入/输出	PA13
JTCK/SWCLK	输入	JTAG 时钟	输入	串行时钟	PA14
JTDI	输入	JTAG 数据输入	——	——	PA15
JTDO/TRACESWO	输出	JTAG 数据输出	——	跟踪时为 TRACESWO 信号	PB3
JNTRST	输入	JTAG 模块复位	——	——	PB4

24.4.2 灵活的 SWJ-DP 脚分配

复位(SYSRESETn 或 PORESETn)以后，属于 SWJ-DP 的所有 5 个引脚都立即被初始化为可被调 试器使用的专用引脚(注意，并没有初始化跟踪输出脚，除非调试器对此脚进行定义)。

然而，HK32F10xxx 微控制器可以用复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)寄存器(见 9.4.2 节)来禁止 SWJ-DP 接口的部分或所有引脚的功能，这些专用引脚将被释放以用作普通 I/O 口。此寄存器被映射到和 Cortex™-M3 系统总线相连接的 APB 桥上。对此寄存器的设置将由用户代码而不是调试器完成。

3 个控制位用来配置 SWJ-DP 接口的引脚，这 3 个位在系统复位时复位。

- AFIO_MAPR(HK32F10xxx 微控制器中的地址是 0x40010004)
  - 读：APB，无等待状态
  - 写：APB，如果 AHB-APB 桥的写缓冲器满了，则一个等待状态位 26:24=SWJ_CFG[2:0] 由软件置位和复位

这 3 位用来设置分配给 SWJ 调试接口的专用引脚数目，目的是在使用不同的调试接口时能释放尽可能多的引脚用作普通 I/O 口。

复位后的初始值是 000(所有引脚都设置为 JTAG-DP 接口专用引脚)，同时只能置位 3 个位中的一个(禁止同时设置一个以上的位)。

表 94 灵活的 SWJ_DP 引脚分配

SWJ-CFG[2:0]	配置为调试专用的引脚	SWJ 接口的 I/O 口分配				
		PA13/ JTMS/ SWDIO	PA14/ JTCK/ SWCLK	PA15/ JTDI	PB3/ JTDO	PB4/ JNTRST
000	所有的 SWJ 引脚 (JTAG-DP + SW-DP) 复位状态	专用	专用	专用	专用	专用
001	所有的 SWJ 引脚 (JTAG-DP + SW-DP) 除了 JNTRST 引脚	专用	专用	专用	专用	释放
010	JTAG-DP 接口禁止，SW-DP 接口允许	专用	专用	释放		
100	JTAG-DP 接口和 SW-DP 接口都禁	释放				
其他	禁止					

注意： *当 APB 桥的写缓冲区满了的时候，在写 AFIO_MAPR 寄存器时需要多用一个 APB 周期。这是因为 JTAGSW 脚的释放需要 2 个 APB 周期，以保证输入内核的 nTRST 和 TCK 信号的平稳。*

- 周期 1：输入 1 / 0 的 JTAGSW 信号到内核(nTRST, TDI 和 TMS 为 1，TCK 为 0)。
- 周期 2：GPI/O 控制器获得 SWJTAG I/O 引脚的控制信号(如对方向，上拉/下拉，施密特触发 等的控制)。

24.4.3 JTAG 脚上的内部上拉和下拉

保证 JTAG 的输入引脚不是悬空的是非常必要的，因为他们直接连接到 D 触发器控制着调试模式。必须特别注意 SWCLK/TCK 引脚，因为他们直接连接到一些 D 触发器的时钟端。

为了避免任何未受控制的 I/O 电平，HK32F10xxx 在 JTAG 输入脚上嵌入了内部上拉和下拉。

- JINTRST：内部上拉
- JTDI：内部上拉
- JTMS/SWDIO:内部上拉
- TCK/SWCLK:内部下拉

一旦 JTAG I/O 被用户代码释放，GPIO 控制器再次取得控制。这些 I/O 口的状态将恢复到复位时的状态。

- JNTRST：带上拉的输入
- JTDI：带上拉的输入
- JTMS/SWDIO：带上拉的输入
- JICK/SWCLK：带下拉的输入
- JTDO：浮动输入

软件可以把这些 I/O 口作为普通的 I/O 口使用。

注意： *JTAG EEE 标准建议对 TDI，TMS 和 nTRST 上拉，而对 TCK 没有特别的建议。但在 HK32F10xxx 中，JTCK 引脚带有下拉。 内嵌的上拉和下拉使芯片不再需要外加外部电阻。*

24.4.4 利用串行接口并释放不用的调试脚作为普通 I/O 口

为了利用串行调试接口来释放一些普通 I/O 口，用户软件必须在复位后设置 SWJ_CFG=010，从而释放 PA15，PB3 和 PB4 用做普通 I/O 口。

在调试时，调试器进行以下操作：

- 在系统复位时，所有 SWJ 引脚被分配为专用引脚(JTAG-DP +SW-DP)。
- 在系统复位状态下，调试器发送指定 JTAG 序列，从 JTAG-DP 切换到 SW-DP。
- 仍然在系统复位状态下，调试器在复位地址处设置断点
- 释放复位信号，内核停止在复位地址处。
- 从这里开始，所有的调试通信将使用 SW-DP 接口，其他 JTAG 引脚可以由用户代码改配为普通 I/O 口。

注意：　对于用户软件设计，应注意：

在复位后，这些专用引脚仍然处于带上拉的输入(nTRST, TMS, TDI)，带下拉的输入(TCK)，和输出(TDO)状态，并持续一段时间，直到用户代码释放这些引脚。

当这些引脚被配置成专用引脚时(JTAG 或者 SW 或者 TRACE)，修改相应的普通 I/O 口配置寄存器是无效的。

24.5 HK32F10xxx JTAG TAP 连接

HK32F10xxx 微控制器内部串联了两个 JTAG TAP。边界扫描 TAP 专门用来进行测试(IR 寄存器为 5 比特位宽)和 Cortex™-M3 TAP(IR 寄存器为有 4 比特位宽)。

为了访问 Cortex™-M3 TAP 对芯片进行调试，必须：

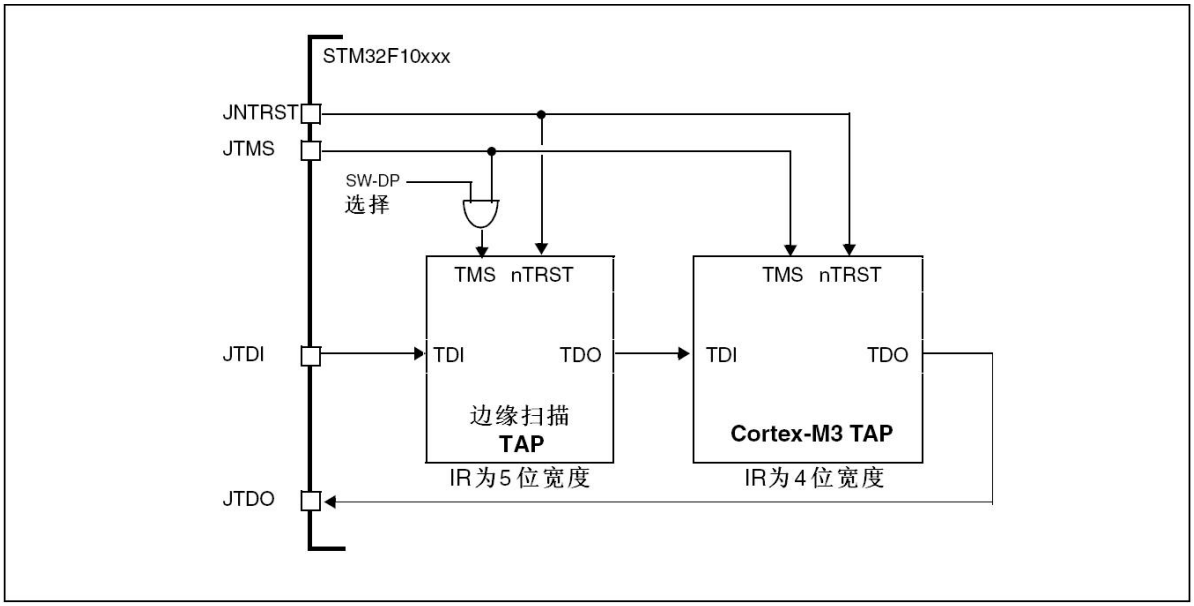
1.首先，必须将 BYPASS 指令移位输入 TMC TAP。

2.其次，在移位输入 IR 时，每个扫描链包含 9 个比特位(=5+4)，对于不用的 TAP，必须输入 BYPASS 指令

3.移位输入数据时，不用的 TAP 处于 BYPASS 模式下，因此数据扫描链需要额外添加一位比特位。

注意：　重要：一旦使用了指定的 JTAG 序列选择了串行调试接口，TMC TAP 自动被禁止(JTMS 被强制为高)。

图 202 JTAG TAP 连接



24.6 ID 代码和锁定机制

在 HK32F10x 微控制器内部有多个 ID 编码。强烈建议工具设计者使用映射在外部 PPB 存储器上地址为 0xE0042000 的 MCU DEVICE ID 来锁定调试器。

24.6.1 微控制器设备 ID 编码

微控制器 HK32F10xxx 内含一个 MCU ID 编码。这个 ID 定义了 MCU 的部件号和硅片版本。它 是 DBG_MCU 的一个组成部分，并且映射到外部 PPB 总线上(见 25.15 节)。使用 JTAG 调试口(4~5 个引脚)或 SW 调试口(2 个引脚)或通过用户代码都可以访问此编码。即使当 MCU 处于系统复位状 态下这个编码也可以被访问。

DBGMCU_IDCODE

地址：0xE004 2000

只支持 32 位访问

只读=0xXXXXX410，其中 X 为内容不确定的位

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				DEV_ID											
				r	r	r	r	r	r	r	r	r	r	r	r

位 31:16	**REV_ID[15:0]**：版本识别 该域标识产品的版本 0x0000 = 版本 A 0x2000 = 版本 B 0x2001 = 版本 Z 0x2003 = 版本 Y
位 15:12	保留
位 11:0	**DEV_ID[11:0]**：设备识别 这个部分指示了设备编码。对于 HK32F10x 微控制器： 设备编码为 0x410；

24.6.2 边界扫描 TAP

JTAG ID 编码

HK32F10xxx 的边界扫描 TAP 集成了 JTAG ID 编码：

- 0x06410041 = 版本 A
- 0x16410041 = 版本 B 和版本 Z

24.6.3 Cortex-M3 TAP

ARM Cortex-M3 的 TAP 有一个 JTAG ID 编码。这个 ID 编码是 ARM 默认的，且没有被修改过，只能通过 JTAG 调试口访问。此编码是 **0x3BA00477**(对应于 Cortex-M3 r1p1)。

调试器/编程工具应该只通过 DEV_ID(11:0)来识别芯片。

24.6.4 Cortex-M3 JEDEC-106 ID 代码

ARM 的Cortex-M3 有一个JEDEC-106 ID 编码。它位于映射到内部PPB 总线地址为0xE00FF000_0xE00FFFFF 的4KB ROM 表中。

24.7 JTAG 调试端口

标准的 JTAG 状态机是通过一个 4 比特位的指令寄存器(IR)和 5 个数据寄存器(详见 *Cortex-M3 r1p1 Technical Reference Manual*)实现的。

表 95 JTAG 调试端口数据寄存器

IR(3:0)	数据寄存器	描述
1111	BYPASS[1 比特位]	
1110	IDCODE[32 比特位]	ID 编码寄存器 0x3BA00477（ARM Cortex-M3 r1p1-01rel0 ID 编码）
1010	DPACC [32 比特位]	调试接口寄存器 初始化调试端口，并允许访问调试接口寄存器 - 输入数据时： Bits34:3=DATA[31:0]：对应写操作的 32 位数据位 Bits2:1=A[3:2]：调试接口寄存器的 2 位地址值 Bit0=RnW：读操作(1)或写操作(0) - 输出数据时： Bits34:3=DATA[31:0]：前一次读操作的 32 位数据结果 Bits2:0=ACK[2:0]：3 比特位的应答 010＝成功/失败001＝等待其他＝未定义 A(3:2)的定义请参考 0
1011	APACC [35 比特位]	存取接口寄存器 初始化存取接口并允许访问存取接口寄存器 - 输入数据时： Bits34:3=DATA[31:0]：对应写操作的 32 位数据位 Bits2:1=A[3:2]：2 比特位地址(AP 寄存器的部分地址) Bit0=RnW：读操作(1)或写操作(0) - 输出数据时： Bits34:3=DATA[31:0]：前一次读操作的 32 位数据结果 Bits2:0=ACK[2:0]：3 比特位的应答 010＝成功/失败001＝等待其他＝未定义 关于 AP 寄存器请参考 AHB-AP 章节，这些寄存器的地址由以下部分组成：A[3:2] - 移位值 A[3:2] - DP SELECT 寄存器的当前值
1000	ABORT [35 比特位]	中止寄存器 - Bits 31:1 未定义 - Bit 0＝DAPABORT：写 1 产生一个 DAP 中止

表 96 由 A[3:2]定义的 32 位调试接口寄存器地址

地址	A(3:2)值	描述
0x0	00	未定义
0x4	01	DP CTRL/STAT 寄存器 - 请求一个系统或调试的上电操作 - 配置 AP 访问的操作模式 - 控制比较，校验操作 - 读取一些状态位(溢出，上电响应)
0x8	10	DP SELECT 寄存器 用来选择当前的访问端口和有效的 4 字长寄存器窗口 - Bits31:24：APSEL 选择当前 AP - Bits23:8：未定义 - Bits7:4：APBANKSEL：在当前 AP 上选择 4 字长寄存器窗口 - Bits3:0：未定义
0xC	11	DP RDBUFF 寄存器：用来使调试器获得前一次操作的最终结果(不用再请求一个新的 JTAG-DP 操作)

24.8 SW 调试端口

24.8.1 SW 协议介绍

此同步串行协议使用 2 个引脚：

- SWCLK：从主机到目标的时钟信号
- SWDIO：双向数据信号

协议允许读写 2 个寄存器组(DPACC 和 APACC 寄存器组)。

数据位按 LSB 传输。

由于 SWDIO 为双向口，该引脚需有上拉(ARM 建议使用 100KΩ电阻)。

按协议每次 SWDIO 方向改变时，需插入一个转换时间。在该期间内主机和目标都不驱动此信号线。转换时间的默认值是 1 个比特，但可以通过配置 SWCLK 频率来调节。

24.8.2 SW 协议序列

每个序列由 3 个阶段组成：

1. 主机发送包请求(8 位)
2. 目标发送确认响应(3 位)
3. 主机或目标发送数据(33 位)

表 97 请求包(8 比特位)

比特位	名称	描述
0	起始	必须为 1
1	APnDP	0：访问 DP　　1：访问 AP
2	RnW	0：写请求　　1：读请求
4:3	A(3:2)	DP 或 AP 寄存器的地址(请参考 0)
5	Parity	前面比特位的校验位
6	Stop	0
7	Park	不能由主机驱动，由于有上拉，目标永远读为 1

有关 DPACC 和 APACC 寄存器描述的详细资料，请参考 Cortex-M3 r1p1 技术参考手册。

包请求后总是跟一个(默认为 1 位)转换时间，此时主机和目标都不驱动线路。

表 98 ACK 定义(3 比特位)

比特位	名称	描述
0..2	ACK	001：失败　010：等待　100：成功

当 ACK 为失败或等待，或者是一个回复读操作的 ACK，此 ACK 后有一个转换时间。

表 99 传输数据(33 比特位)

比特位	名称	描述
0..31	WDATA/ RDATA	写或读的数据
32	Parity	32 位数据的奇偶校验位

读操作的数据传输操作后有一个转换时间。

24.8.3 SW-DP 状态机(Reset, idle states, ID code)

SW-DP 状态机有一个内部 ID 编码用来识别 SW-DP，它遵守 JEP-106 标准。此 ID 编码是 ARM 默认的编码，值为 **0x1BA01477**(对应于 Cortex-M3 r1p1)。

注意：　在调试器读这个 ID 编码之前，SW-DP 的状态机是不工作的。

- SW-DP 状态机将处于 RESET 状态，在上电复位后，或 DP 从 JTAG 切换到 SWD 后，或有超过 50 个周期的高电平。
- 当状态机处于 RESET 状态时，如果有至少 2 个周期的低电平，状态机将切换到 IDLE 状态。
- 当状态机处于 RESET 状态后，必需首先进入 IDLE 状态，并执行一个读 DP-SW ID 寄存器的操作。否则，调试器在执行其他传输时，只能获得一个失败的 ACK 响应。

更详细的 SW-DP 状态机资料请参考 Cortex-M3 r1p1 技术参考手册和 CoreSight Design Kit r1p0 技术参考手册。

24.8.4 DP 和 AP 读/写访问

- 对 DP 的读操作没有延迟：调试器将直接获得数据(如果 ACK＝成功)，或者等待(如果 ACK=等待)。
- 对 AP 的读操作具有延迟。即前一次读操作的结果只能在下一次操作时获得。如果下一次的操作不是对 AP 的访问，则必需读 DP-RDBUFF 寄存器来获得上一次读操作的结果。
- DP-CTRL/STAT 寄存器的 READOK 标志位会在每次 AP 读操作和 RDBUFF 读操作后更新，以 通知调试器 AP 的读操作是否成功。
- SW-DP 具有写缓冲区(DP 和 AP 都有写缓冲)，这使得其他传输在进行时，仍然可以接受写操 作。如果写缓冲区满，调试器将获得一个等待的 ACK 响应。读 IDCODE 寄存器，读 CTRL/STAT 寄存器和写 ABORT 寄存器操作在写缓冲区满时仍被接受。
- 由于 SWCLK 和 HCLK 的异步性，需要在写操作后(在奇偶校验位后)插入 2 个额外的 SWCLK 周期，以确保内部写操作正确完成。这两个额外的时钟周期需要在线路为低时插入(IDLE 状态下)。这个操作步骤在写 CTRL/STAT 寄存器以提出一个上电请求时尤其重要，否则下一个操作(在内核上电后才有效的操作)会立即执行，这将导致失败。

24.8.5 SW-DP 寄存器

当 APnDP=0 时，可以访问以下这些寄存器。

表 100 SW-DP 寄存器

A(3:2)	读/写	SELECT 寄存器的 CTRLSEL 位	寄存器	描述
00	读		IDCODE	固定为 0x1BA01477(用于识别 SW-DP)。
00	写		ABORT	
01	读/写	0	DP-CTRL/STAT	－请求一个系统或调试的上电操作； －配置 AP 访问的操作模式； －控制比较，校验操作； －读取一些状态位(溢出，上电响应)。
01	读/写	1	WIRE CONTROL	配置串行通信物理层协议(如转换时间长度等)。
10	读		READ RESEND	允许从一个错误的调试传输中恢复数据而不用重复最初的 AP 传输。
10	写		SELECT	选择当前的访问端口和有效的 4 字长寄存器窗口。
11	读/写		READ BUFFER	由于 AP 的访问具有传递性(当前 AP 读操作的结果会 在下次 AP 传输时传出),因此这个寄存器非常必要。 这个寄存器会从 AP 捕获上一次读操作的数据结果，因此可以获得数据而不必再启动一个新的 AP 传输。

24.8.6 SW-AP 寄存器

当 APnDP=1 时，可以访问以下这些寄存器。

AP 寄存器的访问地址由以下两部分组成：

- A[3:2]的值
- DP SELECT 寄存器的当前值

24.9 对于 JTAG-DP 或 SWDP 都有效的 AHB-AP (AHB 访问端口)

功能：

- 系统访问是独立于处理器状态的。
- JTAG-DP 和 SW-DP 都可以访问 AHB-AP
- AHB-AP 是总线矩阵的 AHB 主设备。因此,它可以访问所有的数据总线(Dcode 总线，System 总线，内部和外部 PPB 总线)，只有 ICode 总线除外。
- 支持位寻址的传输
- 旁路 FPB 的 AHB-AP 传输

32 位 AHP-AP 寄存器的地址是 6-位宽(最多 64 个字或 256 个字节)，由以下部分组成：

a)比特位[8:4]= DP SELECT 寄存器的位[7:4] APBANKSEL

b)比特位[3:2] = 35 位SW-DP 包请求中的A(3:2)。

Cortex-M3 的 AHB-AP 有 9 个 32 位的寄存器

表 101 Cortex-M3 AHB-AP 寄存器

地址偏移	寄存器名	描述
0x00	AHB-AP Control and Status Word	配置 AHB 接口的传输特性(长度，地址自加模式，当前传输状态，特权模式等)。
0x04	AHB-AP Transfer Address	
0x0C	AHB-AP Data Read/Write	
0x10	AHB-AP Banked Data 0	直接访问 4 个相连的字而不用重写访问地址。
0x14	AHB-AP Banked Data 1	
0x18	AHB-AP Banked Data 2	
0x1C	AHB-AP Banked Data 3	
0xF8	AHB-AP Debug ROM Address	调试接口的基地址。
0xFC	AHB-AP ID Register	

更多信息请参考 Cortex-M3 r1p1 技术参考手册

24.10 内核调试

通过操作内核调试寄存器可以实行对内核的调试。对这些寄存器的访问通过先进高性能总线 (AHB-AP)进行。处理器可以通过内部私有外设总线(PPB)直接访问这些寄存器。

它包括 4 个寄存器。

表 102 内核调试寄存器

寄存器	描述
DHCSR	32 位的调试控制和状态寄存器 此寄存器提供内核状态信息，允许内核进入调试模式，和提供单步功能。
DCRSR	17 位的内核寄存器调试选择寄存器 此寄存器选择需要进行读写操作的内核寄存器。
DCRDR	32 位的内核寄存器调试数据寄存器 此寄存器存放由 DCRSR 选择的内核寄存器读出的或需要写入的数据。
DEMCR	32 位异常调试和监视控制寄存器 此寄存器提供向量传输和监视调试控制功能。TRCENA 位启动 TRACE 功能。

注意： ***重要****：这些寄存器在系统复位时不复位，仅在上电复位时复位。*

更多详细资料请参考 Cortex-M3 r1p1 技术参考手册。

为了在复位后立即使内核进入调试状态，需要：

- 使能调试和异常监视控制寄存器(Debug and Exception Monitor Control Register)的位 0 (VC_CORRESET)。
- 使能调试控制和状态寄存器(Debug Halting Control and Status Register)的位 0 (C_DEBUGEN)。

24.11 调试器主机在系统复位下的连接能力

HK32F10xxx 微控制器的复位系统由下列复位源组成：

- POR(上电复位)，在每次上电时发起一次复位
- 内部看门狗复位
- 软件复位
- 外部复位

Cortex-M3 将调试部分的复位(通常是 PORRESETn)和其他复位(SYSRESETn)区分开。因此，

当内核处于系统复位状态时，调试器可以连接到内核，配置内核调试寄存器，使能调试允许 位，这样操作使内核在系统复位被释放时立即进入调试状态而不执行任何指令。同样的，可以在内核处于复位状态下时配置调试特性。

注意：　强烈建议调试器在系统复位时连接内核(在复位向量处设置断点)。

24.12 FPB (Flash patch breakpoint)

FPB 单元：

- 实现硬件断点
- 用系统区域的代码和数据取代代码区域的代码和数据。此特性可以用来纠正代码区域内的软件错误。

软件补丁功能和硬件断点功能不能同时使用。

FPB 由以下部分组成：

- 2 个内容比较器，用来比较代码区域取得的内容并重映射到系统区域的相关地址。
- 6 个指令比较器，用来比较代码区域的指令。这些比较器可用来实现软件补丁或者硬件断点功能。

24.13 DWT(数据观察点触发 data watchpoint trigger)

DWT 模块由四个比较器组成，它们分别是：

- 一个硬件数据比较器
- 一个 PC 值取样器
- 一个数据地址取样器

DWT 还可用来获取某些侧面的信息。通过一些计数器可以获得以下数据：

- 时钟周期
- 分支指令
- 存取单元操作
- 睡眠周期
- CPI(每条指令的执行时间)
- 中断开销

24.14 ITM (指令跟踪微单元 instrumentation trace macrocell)

24.14.1 概述

ITM 是一应用驱动的跟踪源，它支持 *printf* 类的调试手段来跟踪操作系统(OS)和应用事件，并发布判定的系统信息。ITM 以包的形式发布跟踪信息，它由以下部分组成：

- 软件跟踪：软件可以通过直接写 ITM 激发寄存器来发布包信息。
- 硬件跟踪：ITM 会发布由 DWT 产生的信息包。
- 时间戳：时间戳被发布到相应的包上。ITM 包含一个 21 位的计数器以产生时间戳。Cortex-M3 的时钟或串行线观测器(*Serial Wire Viewer*)的位时钟率给计数器提供时钟。

由 ITM 发送的信息包输出到 TPIU(Trace Port Interface Unit)，TPIU 再添加一些额外的包(参考TPIU)，然后输出完整的包序列给调试器。

用户在设置或使用 ITM 之前，必需先使能异常调试和监视控制寄存器(Debug Exception andMonitor Control Register)的 TRCEN 位。

24.14.2 时间戳包，同步和溢出包

时间戳包包含了时间戳信息，普通的控制和同步信息。它使用一个 21 位的时间戳计数器(及可能的预分频器)，此计数器在每个时间戳包发放时复位。计数器的时钟可以是 CPU 时钟也可以

是SWV 时钟。

同步包为 0x80_00_00_00_00_00，按 00 00 00 00 00 80 发送给 TPIU(LSB 在前)。 同步包是时间戳包的控制信号。

它也在每个 DWT 触发时发送，因此 DWT 必须配置为触发 ITM：必须设置 DWT 控制寄存器(DWT Control Register)的位 0(CYCCNTENA)。此外，也必须设置 ITM 跟踪控制寄存器(Trace Control Register)的位 2(SYNCENA)。

注意： *如果 SYNENA 位没有被置起，DWT 产生给 TPIU 的同步触发，将只发送 TPIU 同步包而不发送 ITM 同步包。*

溢出包是一个特殊的时间戳包，该包指示数据已经被写但是 FIFO 已满。

表 103 主要的 ITM 寄存器

地址	寄存器	描述
@E0000FB0	ITM Lock Access	写入 0xC5ACCE55 允许写其他 ITM 寄存器
@E0000E80	ITM Trace Control	位 31-24 = 总是 0
		位 23 = 忙 位 22-16 = 7 位的 ATB ID 用以识别跟踪数据源 位 15-10 = 总是 0 位 9:8 = 时间戳的预分频 位 7-5 = 未定义 位 4 = 使能 SWV 功能即时间戳计数器使用 SWV 时钟 位 3 = 使能 DWT 的激发功能 位 2 = 此位必需设为 1 来使能 DWT 的产生同步触发功能，使 TPIU 能够发送同步包 位 1 = 时间戳使能 位 0 = ITM 的全局使能位
@E0000E40	ITM Trace Privilege	位 3：置'1'使能跟踪端口 31:24 位 2：置'1'使能跟踪端口 23:16 位 1：置'1'使能跟踪端口 15:8 位 0：置'1'使能跟踪端口 7:0
@E0000E00	ITM Trace Enable	每个比特位使能相应的触发端口产生跟踪
@E0000000－E000007C	Stimulus Port Registers 0-31	向选中的产生跟踪的触发端口(32 个)写 32 位数据

关于配置的例子：

向 TUIU 输出一个简单值：

- 配置 TPIU 并使能 I/IO_TRACEN 以使 MCU 分配 TRACE 的引脚(参见 25.16.2 节-跟踪引脚分配，25.15.3 节-调试 MCU 配置寄存器)；
- 向 ITM Lock Access 寄存器写入 0xC5ACCE55，以允许写其他 ITM 寄存器；
- 向 Trace Control 寄存器写入 0x00010005，使能 TPIU 的同步包并使能整个 ITM 功能，寄存器 中的 ATB ID 为 0x01；
- 向 ITM Trace Enable 寄存器写入 0x1，以使能触发端口 0；
- 向 ITM Trace Privilege 寄存器写入 0x1，关闭对触发端口 7:0 的屏蔽；
- 把需要输出的值写入触发端口 0 寄存器，这个步骤可以通过软件完成(使用 printf 功能)。

24.15 MCU 调试模块(MCUDBG)

MCU 调试模块协助调试器提供以下功能：

- 低功耗模式
- 在断点时提供定时器、看门狗、I^2C 和 bxCAN 的时钟控制
- 对跟踪脚分配的控制

24.15.1 低功耗模式的调试支持

使用 WFI 和 WFE 可以进入低功耗模式。

MCU 支持多种低功耗模式，分别可以关闭 CPU 时钟，或降低 CPU 的能耗。

内核不允许在调试期间关闭 FCLK 或 HCLK。这些时钟对于调试操作是必要的，因此在调试期间，它们必须工作。MCU 使用一种特殊的方式，允许用户在低功耗模式下调试代码。
为实现这一功能，调试器必须先设置一些配置寄存器来改变低功耗模式的特性。

- 在睡眠模式下，调试器必须先置位 DBGMCU_CR 寄存器的 DBG_SLEEP 位。这将为 HCLK 提供与 FCLK(由代码配置的系统时钟)相同的时钟。
- 在停止模式下，调试器必须先置位 DBG_STOP 位。这将激活内部 RC 振荡器，在停止模式下 为 FCLK 和 HCLK 提供时钟。

24.15.2 支持定时器、看门狗、bxCAN 和 I2C 的调试

在产生断点时，有必要根据定时器和看门狗的不同用途选择计数器的工作模式：

- 在产生断点时，计数器继续计数。这在输出 PWM 控制电机时常常要用到。
- 在产生断点时，计数器停止计数。这对于看门狗的计数器是必需的。

对于 bxCAN，用户可以选择在断点期间阻止接收寄存器的更新。 对于 I^2C，用户可以选择在断点期间阻止 SMBUS 超时。

24.15.3 调试 MCU 配置寄存器

此寄存器允许在调试状态下下配置 MCU。包括：

- 支持低功耗模式
- 支持定时器和看门狗的计数器
- 支持 bxCAN 通信
- 分配跟踪引脚

DBGMCU_CR 寄存器被映射到外部 PPB 总线，基地址为 0xE0042000。

寄存器由 PORESET 异步复位(不被系统复位所复位)。当内核处于复位状态下时，调试器可写该寄存器。

如果调试器不支持这些特性，用户软件仍可写这些寄存器。

DBGMCU_CR

地址：0xE0042004

只支持 32 位访问

POR 复位：0x0000 0000(不被系统复位所复位)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															DBG_I2C2_SMBUS_TIMEOUT
res										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_I2C1_SMBUS	DBG_CAN_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP	DBG_TIM1_STOP	DBG_WWDG_STOP	DBG_IWDG_STOP	TRACE_MODE[1:0]		TRACE_IOEN	保留		DBG_STANDBY	DBG_STOP	DBG_SLEEP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res		rw	rw	rw

位 31:22	保留，必须保持为 0。
位 21:17	保留
位 16	**DBG_I2C2_SMBUS_TIMEOUT**：当核心停止时停止 SMBUS 超时模式。 0: 与正常模式操作相同； 1: 冻结 SMBUS 的超时控制。
位 15	**DBG_I2C1_SMBUS_TIMEOUT**：当核心停止时停止 SMBUS 超时模式。 0: 与正常模式操作相同； 1: 冻结 SMBUS 的超时控制。
位 14	**DBG_CAN_STOP**：当内核进入调试状态时，CAN 停止运行。 0：CAN 仍然正常运行； 1：CAN 的接收寄存器不继续接收数据。
位 13:10	**DBG_TIMx_STOP**：当内核进入调试状态时计数器停止工作x=4..1。 0：选中定时器的计数器仍然正常工作； 1：选中定时器的计数器停止工作。

位 9	**DBG_WWDG_STOP**：当内核进入调试状态时调试窗口看门狗停止工作。 0：窗口看门狗计数器仍然正常工作； 1：窗口看门狗计数器停止工作。
位 8	**DBG_IWDG_STOP**：当内核进入调试状态时看门狗停止工作 0：看门狗计数器仍然正常工作； 1：看门狗计数器停止工作。
位 7:5	**TRACE_MODE[1:0]**和 **TRACE_IOEN**：跟踪引脚分配控制 - 当 TRACE_IOEN＝0 时：TRACE_MODE＝xx：不分配跟踪引脚(默认状态)。 - 当 TRACE_IOEN=1 时： TRACE_MODE=00：跟踪引脚使用异步模式； TRACE_MODE=01：跟踪引脚使用同步模式，并且数据长度为 1； TRACE_MODE=10:跟踪引脚使用同步模式，并且数据长度为 2； TRACE_MODE=11:跟踪引脚使用同步模式，并且数据长度为 4。
位 4:3	保留，必须保持为 0。
位 2	**DBG_STANDBY**：调试待机模式。 0：(FCLK 关，HCLK 关)整个数字电路部分都断电。 从软件的观点看，退出 STANDBY 模式与复位是一样的(除了一些状态位指示了微控制器刚从 STANDBY 状态退出)。 1：(FCLK 开，HCLK 开)数字电路部分不下电，FCLK 和 HCLK 时钟由内部 RL 振荡器提供时钟。 另外，微控制器通过产生系统复位来退出 STANDBY 模式和复位是一样的。
位 1	**DBG_STOP**：调试停止模式。 0：(FCLK 关，HCLK 关)在停止模式时，时钟控制器禁止一切时钟(包括 HCLK 和 FCLK)。当从 STOP 模式退出时，时钟的配置和复位之后的配置一样(微控制器由 8MHz 的内部 RC 振荡器(HIS)提供时钟)。因此，软件必需重新配置时钟控制系统启动 PLL，晶振等。 1：(FCLK 开，HCLK 开)在停止模式时，FCLK 和 HCLK 时钟由内部 RC 振荡器提供。当退出停止模式时，软件必需重新配置时钟系统启动 PLL，晶振等(与配置此比特位为 0 时的操作一样)。
位 0	**DBG_SLEEP**：调试睡眠模式 0：(FCLK 开，HCLK 关)在睡眠模式时，FCLK 由原先已配置好的系统时钟提供，HCLK 则关闭。由于睡眠模式不会复位已配置好的时钟系统，因此从睡眠模式退出时，软件不需要重新配置时钟系统。 1：(FCLK 开，HCLK 开)在睡眠模式时，FCLK 和 HCLK 时钟都由原先配置好的系统时钟提供。

24.16 TPIU (跟踪端口接口单元 Trace Port Interface Unit)

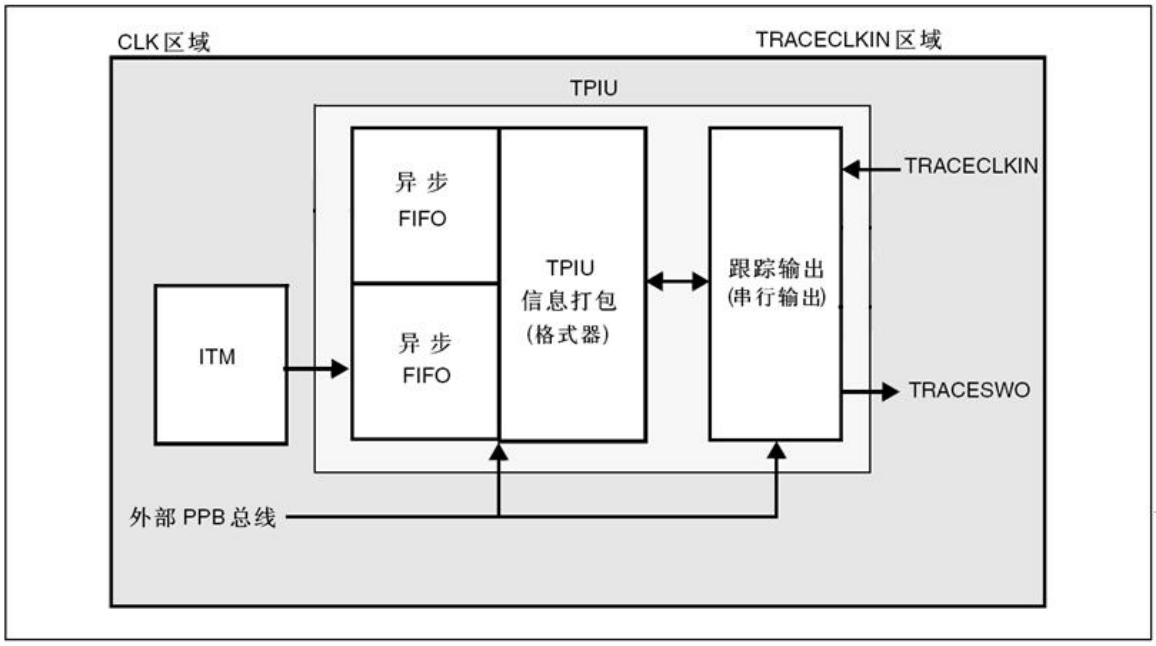
24.16.1 导言

TPIU 在片上数据跟踪和 ITM 之间担当桥梁的作用。

输出的数据流封装成跟踪源 ID，然后被追踪端口分析器(*Trace Port Analyzer*)采集。

内核嵌入了一个简单的专门为低价调试所设计的 TPIU(由一个特殊版本的 CoreSight TPIU 组成)。

图 203 TPIU 框图



24.16.2 跟踪引脚分配

- 异步模式

异步模式需要 1 个额外的引脚并且存在于所有的封装。此模式仅在串行调试接口有效(不支持 JTAG 调试接口)。

表 104 异步跟踪引脚分配

TPUI 引脚	同步跟踪模式		HK32F10xxx 引脚分配
	类型	描述	
TRACESWO	输出	异步跟踪数据输出	PB3

TPUI 跟踪引脚分配

这些引脚在默认状态下不是专用引脚。可以通过设置 *MCU Debug Component Configuration* 寄存器的 TRACE_IOEN 和 TRACE_MODE 位分配这些引脚。必需由调试器完成设置。

此外，由跟踪的配置决定分配的引脚数(异步)。

- 异步模式：需要 1 个额外引脚

调试器需要设置 Debug MCU Configuration 寄存器的 TRACE_IOEN 和 TRACE_MODE[1:0]位来分配跟踪引脚。默认时，跟踪脚是不分配的。

此寄存器被映射到外部 PPB 并且被 PORESET 所复位(系统复位不复位此寄存器)。调试器可以

在系统复位的状态下写该寄存器。

表 105 灵活的跟踪引脚分配

DBGMCU_CR 寄存器		引脚用途	跟踪引脚分配
TRACE_IOEN	TRACE_MODE[1:0]		PB3/ JTDO/ TRACES WO
0	XX	无跟踪 (默认状态)	释放(1)
1	00	异步跟踪	TRACES WO
1	01	保留	释放(1)
1	10	保留	
1	11	保留	

注意： *TUIP 的输入时钟 TRACECLKIN 默认接地。所以在比特位 TRACE_IOEN 被置位后，HCLK 需要两个时钟周期。*

然后，调试器可以通过写 TPIU 的 SPP_R(Selected Pin Protocol)寄存器的 PROTOCOL [1:0]位来配置跟踪模式。

- PROTOCOL=01 或 10：串行模式(曼彻斯特或 NRZ 编码)。默认状态为 01 然后通过写 TPIU 的 CPSPS_R(Current Sync Port Size)寄存器的位[3:0]来配置跟踪端口的大小。
- 0x1：1 个引脚(默认)
- 0x2：2 个引脚
- 0x8：4 个引脚

24.16.3 TPUI 格式器

协议格式器输出 16 个字节组成的帧：

- 7 个数据字节
- 8 个多用途字节，由以下部分组成：
  - 1 位(LSB)用来区分数据字节(0)和 ID 字节(1)。
  - 7 位(MSB)可以作为数据或跟踪源 ID 的变化。
- 1 个辅助字节，其中的每个位都对应于 8 个多用途字节中的一个：
  - 如果对应的多用途字节是数据字节，那么这个位是数据的比特 0 位。
  - 如果对应字节是 ID 字节，这个位表明 ID 变化何时生效。

注意：　更多信息，请参考 ARM CoreSight Architecture Specification v1.0(ARM IHI 0029B)

24.16.4 TPUI 帧异步包

TPUI 会产生两种类型的同步包：

- 帧同步包(或全字同步包) 该包为 0x7F_FF_FF_FF(LSB 先发)，这个序列只有在 0x7F 没有被作为 ID 源编码的时候才能使用。
  该包在帧之间周期性地输出。 在连续模式里，一旦同步帧被发现，TPA 必须抛弃所有这些帧。
- 半字同步包 该包为：0x7F_FF(LSB 先发)。
  它在帧之间或帧内周期性的输出。
  这些包只存在于连续模式中，并且使能 TPA 检测 IDLE 模式下的 TRACE 口(无 TRACE 被捕捉)。当被 TPA 检测到时，必须将其抛弃。

24.16.5 异步模式

调试模块提供一个低成本的，只使用一个引脚的跟踪数据输出功能，即使用异步输出引脚 TRACESWO。但显然，这样的输出数据带宽是有限的。 TRACESWO 引脚与 JTDO 引脚复用，只在 SW-DP 调试接口有效，因此 HK32F10xxx 的所有封
装都提供这种功能。

异步模式需要 TRACECLKIN 引脚有平稳的频率提供。对标准的 UART(NRZ)捕捉机制来说，需要
5%的正确度。曼彻斯特编码可放宽到 10%。

24.16.6 TRACECLKIN 在 HK32F10xxx 内部的连接

TRACECLKIN 输入在 HK32F10xxx 内部与 HCLK 相连接。这意味着在使用异步跟踪模式时，应用程序应限制使用时间帧保证 CPU 频率的稳定。

注意：　重要：当使用异步跟踪功能时需注意：

HK32F10xxx 微控制器的初时时钟是内部 RC 振荡器，此振荡器在复位状态下的频率与复位后 的频率不同。这是由于 RC 校准在复位状态下使用初始值，而这个值在复位释放后会更新。 因此，不应该在系统复位状态下激活跟踪端口分析器(Trace Port Analyzer)的跟踪功能(置位 TRACE_IOEN)。因为在复位状态下的同步帧包的比特宽度与复位后的包不同。

24.16.7 TPIU 寄存器

只有当 Debug Exception and Monitor Control(DEMCR)寄存器的 TRCENA 位被置位时，TPIU APB 寄存器才可以被读写。否则寄存器读出值为 0(这一位的输出使能 TPIU 的 PCLK)。

表 106 重要的 TPIU 寄存器

地址	寄存器	描述
0xE0040004	Current port size	跟踪端口的长度： 位 0：端口长度为 1 位 1：端口长度为 2 位 2：端口长度为 3，不支持 位 3：端口长度为 4 四个比特中只能同时置位一个比特。 默认状态下，端口长度为 1(0x00000001)
0xE00400F0	Selected pin protocol	跟踪端口协议的选择： 位 1:0＝ 00：同步跟踪模式 01：串行输出－曼彻斯特编码(默认值) 10：串行输出－NRZ 11：未定义
0xE0040304	Formatter and flush control	位 31-9 ：总是 0 位 8 ＝TrigIn：总是 1，指示触发器 位 7-4：总是 0 位 3-2：总是 0 位 1＝EnFCont：同步模式下(Select Pin Protocol 寄存器的 Bit1：0 为 00)，此比特位强制为 1，连续模式下格式器被自动使能。异步模式下(Select Pin Protocol 寄存器的 Bit1：0 不为 00)，此比特可以被置位或复位来选择是否使能格式器。 位 0：总是 0 默认值为 0x102 注意：在同步模式下，由于 TRACECTL 信号没有外部引脚，因此格式器会在连续模式下自动使能。这意味着格式器会插入一些控制包来识别跟踪包的源。
0xE0040300	Formatter and flush status	没有在 Cortex-M3 中使用，读出值始终为 0x00000008

24.16.8 配置的例子

- 设置 Debug Exception and Monitor Control 寄存器的 TRCENA 位；
- 在 TPIU Current Port Size 寄存器中写入期望值(默认是 0x1，指示端口长度为 1bit)；
- 向 TPIU Formatter and Flush Control 寄存器中写入 0x102(默认值)；
- 写 TPIU Select Pin Protocol 寄存器，选择异步模式。例如写 0x2 选择 NRZ 编码的异步模式(类似 URAT)；
- 向 DBGMCU Control 寄存器写入 0x20(置位 IO_TRACEN)，为异步模式分配 TRACE 的 I/O 口。此时 TPIU 将发出一个同步包(FF_FF_FF_7F)；
- 配置 ITM 并且写 ITMStimulus 寄存器输出数据

24.17 DBG 寄存器地址映象

下列表格归纳了调试寄存器。

表 107 DBG – 寄存器和复位值

地址	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE0042000	DBGMCU_IDCODE	REV_ID																保留				DEV_ID											
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x	x	x	x	x	x	x	x	x
0xE0042004	DBGMCU_CR	保留										-	-				DBG_I2C2_SMBUS_TIMEOU	DBG_I2C1_SMBUS_TIMEOU	DBG_CAN_STOP	DBG_TIM4_STP	DBG_TIM3_STP	DBG_TIM2_STP	DBG_TIM1_STP	DBG_WWDG_STOP	DBG_IWDG_STOP	TRACE_MODE [1:0]		TRACE_IOEN	保留		DBG_STANDBY	DBG_STOP	DBG_SLEEP
	复位值												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						

(1) 复位值与特定的产品相关。详情参见 25.6.1 节-微控制器设备 ID 编码

25 重要提示

在未经深圳市航顺芯片技术研发有限公司同意下不得以任何形式或途径修改本公司产品规格和数据表中的任何部分以及子部份。深圳市航顺芯片技术研发有限公司在以下方面保留权利：修改数据单和/或产品、停产任一产品或者终止服务不做通知；建议顾客获取最新版本的相关信息，在下定订单前进行核实以确保信息的及时性和完整性。所有的产品都依据订单确认时所提供的销售合同条款出售，条款内容包括保修范围、知识产权和责任范围。

深圳市航顺芯片技术研发有限公司保证在销售期间，产品的性能按照本公司的标准保修。公司认为有必要维持此项保修，会使用测试和其他质量控制技术。除了政府强制规定外，其他仪器的测量表没有必要进行特殊测试。

顾客认可本公司的产品的设计、生产的目的不涉及与生命保障相关或者用于其他危险的活动或者环境的其他系统或产品中。出现故障的产品会导致人身伤亡、财产或环境的损伤（统称高危活动）。人为在高危活动中使用本公司产品，本公司据此不作保修，并且不对顾客或者第三方负有责任。

深圳市航顺芯片技术研发有限公司将会提供与现在一样的技术支持、帮助、建议和信息，（全部包括关于购买的电路板或其他应用程序的设计，开发或调试）。特此声明，对于所有的技术支持、可销性或针对特定用途，及在支持技术无误下，电路板和其他应用程序可以操作或运行的，本公司将不作任何有关此类支持技术的担保，并对您在使用这项支持服务不负任何法律责任。